VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Paketový filtr v prostředí Linux

Linux packet filter

2017

Bc. Michal Czyž, DiS.

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Michal Czyž, DiS.**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Paketový filtr v prostředí Linux
Linux packet filter

Jazyk vypracování: čeština

Zásady pro vypracování:

Operační systém Linux obsahuje mnoho nástrojů a technologií, které slouží k zabezpečení síťového provozu. Jedním z nástrojů je i paketový filtr, který je součástí linuxového jádra. Paketový filtr kontroluje hlavičky příchozích a odchozích paketů a podle jejího obsahu rozhoduje, jak se s každým jednotlivým paketem naloží, zda bude paket akceptován, zahozen a podobně. Cílem diplomové práce je ochrana zdrojů a dat ve své vlastní počítačové sítí za použití paketového filtru.

Diplomová práce bude obsahovat:
- Teoretický rozbor paketového filtru a jejich porovnání
- Průniky
- Nastavení paketového filtru
- Vyhodnocení výsledků, použitých metod a jejich přínosů.

Seznam doporučené odborné literatury:

- Bob Toxen: Bezpečnost v Linuxu - Prevence a odvracení napadení systému, 2003, Computer Press, ISBN: 80-7226-716-7
- Matúš Selecký: Penetrační testy a exploitace, 2012 (1. vydání), Computer Press, ISBN: 978-80-251-3752-9
- Daniel J. Barrett, Richard E. Silverman, Robert G. Byrnes: Linux Security Cookbook, 2003, O'Reilly Media, ISBN: 978-0596003913
- Peter G Smith, Linux Network Security, 2005, Charles River Media, ISBN: 978-1584503965
- Tony Bautts, Terry Dawson, Gregor N. Purdy: Linux Network Administrator's Guide, 2005, O'Reilly Media, 3rd Edition, ISBN: 978-0-59600-548-1

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Jaroslav Zdrálek, Ph.D.**

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *15. dubna 2017*

.....
podpis studenta

Poděkování

Rád bych poděkoval doc. Ing. Jaroslavu Zdrálkovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této diplomové práce.

Abstrakt

Tato diplomová práce má za cíl vytvoření stavového paketového filtru pomocí iptables, který je jedním z nástrojů, sloužící k filtraci síťového provozu a zabezpečení sítě v prostředí operačního systému Linux. V první části práce je rozebrána základní charakteristika protokolu TCP/IP, společně s popisem zvolené distribuce operačního systému a nástroje pro jeho virtualizaci. Je rovněž uveden popis generačního rozdělení zařízení, které filtrují síťový provoz a to včetně problematiky frameworku Netfilter a nástroje iptables. Druhá část práce souvisí se základním nastavením výchozí bezpečnostní politiky pomocí iptables, včetně popisu jednotlivých částí. Dále pak charakteristikou smyslu etického hackingu, výběrem penetračního nástroje a použití nástroje Cacti za účelem monitorování specifických parametrů v rámci vytvořené sítě. V této části bude rovněž provedeno testování zvolené bezpečnostní politiky včetně popisu souvisejících výsledků. Třetí část práce se soustředí na simulaci tří vybraných typů síťových útoků na nezabezpečenou a zabezpečenou síť. Rozpoznání jejich charakteristických znaků a následné potlačení či omezení útoku za použití navrhnutých bezpečnostních pravidel. Část zahrnuje rovněž analýzu informací zaznamenaných během testů. V závěru práce jsou shrnuty cíle, kterých bylo dosaženo.

Klíčová slova

LINUX, VIRTUALIZACE, IPTABLES, NETFILTER, IP, DOS, DDOS, BEZPEČNOSTNÍ POLITIKA, TCP, UDP, STAVOVÝ PAKETOVÝ FILTR

Abstract

This diploma thesis aims at creating a stateful packet filter using iptables, which is one of the tools used to filter network traffic and improve network security in a Linux environment. The first thesis part describes the basic characteristics of TCP/IP protocol, together with a description of the chosen distribution of the operating system and virtualization tool. It is also introduced a generational division description of the devices that filter network traffic, including the problematics of framework Netfilter and iptables tool. The second thesis part is relating to the basic settings of the default security policy using iptables, including a description of the individual parts. Further characteristic meaning of ethical hacking, choice of penetration tool and use the Cacti tool for a purpose to monitor certain parameters in the network. In this section will also carried out by the testing of created security policy, including a description of the related results. The third thesis part focuses on the simulation of three selected types of network attacks on unsecured and secured network. Recognizing their characteristics and the subsequent suppression or limitation of the attack using the proposed security rules. This part also includes a detailed analysis of the information recorded during the test. The end of the thesis summarizes achieved goals.

Key words

LINUX, VIRTUALIZATION, IPTABLES, NETFILTER, IP, DOS, DDOS, SECURITY POLICY, TCP, UDP, STATEFUL PACKET FILTER

Obsah

Seznam použitých symbolů

Symbol	Jednotky	Význam symbolu
Přenosová rychlost	bit/s	Objem informace přenesené za jednotku času
Čas	s	Základní veličina soustavy SI

Seznam použitých zkratek

Zkratka	Význam
CPU	Procesor
DoS	Odepření služby
DDoS	Distribuované odepření služby
ICMP	Protokol používající zasílání kontrolních zpráv
IP	Internetový protokol
ISO	Mezinárodní organizace zabývající se tvorbou norem
LAN	Označení pro lokální síť
OSI	Standardizace v komunikacích
RAM	Paměť s libovolným přístupem
TCP	Spolehlivý transportní protokol
TCP/IP	Sada protokolů pro komunikaci v síti
UDP	Nespolehlivý transportní protokol
MAC	Jednoznačný identifikátor síťového zařízení
ARP	Protokol k získání linkové adresy síťového rozhraní
MTU	Maximální přenosová jednotka
NAT	Překlad síťových adres
TTL	Omezuje maximální dobu existence datagramu
OS	Operační systém
SSH	Zabezpečený komunikační protokol
HTTP	Protokol určený pro výměnu hypertextových dokumentů
HTTPS	Protokol určený pro zabezpečenou výměnu hypertextových dokumentů
GNU	Svobodný operační systém projektu GNU

Seznam ilustrací a seznam tabulek

Úvod

Bezpečnost počítačů, sítí a informačních systémů je bezesporu důležitým tématem. Nedostatečné zabezpečení těchto systémů nebo špatně nastavená bezpečnostní politika umožňuje případnému útočníkovi neoprávněný přístup a může vést ke zneužití. V současné době narůstá četnost síťových útoků za cílem odcizení citlivých osobních údajů z korporací či vládních agentur, ale také přímo na data jednotlivých uživatelů, které mají uložena na svých počítačích. Dosažení úplné ochrany před potencionálními útočníky se tak stává věčnou výzvou, jelikož neexistuje žádná dostatečná technologie navržena tak, aby zcela a úspěšně ochránila samotný systém a počítačovou síť. Jak postupem času neustále vznikají nové bezpečnostní hrozby, je potřebné držet s útočníky krok a přizpůsobovat tomu metody zajištění ochrany naší sítě. To zvláště platí v oblastech systémů detekce síťových narušení a zařízení k řízení přístupu do sítě, tedy paketový filtr (firewall). V operačním systému Linux je možnou alternativou k vytvoření paketového filtru kombinace nástrojů Netfilter/iptables. Linux tak představuje spolehlivé a levné řešení, které je navíc plně přizpůsobitelné požadavkům síťového administrátora.

Důvod, proč byl vybrán zrovna tento operační systém, je zjevně ten, že se začíná používat v čím dál větším měřítku. Linux je nasazen v mnoha sítích jako velmi důležitý prvek a pro funkčnost tohoto celku je mnohdy nezbytný. Proto je nutné, aby byl dobře chráněn on samotný a pokud se používá jako server k oddělení internetu a intranetu, tak také aby chránil tyto dva celky mezi sebou. Na základě této skutečnosti se stává síť, kterou provozujeme, v potencionálním nebezpečí že bude napadena a ohrožena každou vteřinu dne. Striktní nasazením síťové filtrovací politiky pomocí iptables, je vhodným prvním krokem k udržení silného bezpečnostního postoje. Rizika spojené se ztrátou dat či omezení funkčnosti či chodu systému pravděpodobně značně převažují náklady potřebné pro nasazení a udržování iptables v systému Linux.

Tato diplomová práce je rozdělena do tří částí. První část rozebírá základní charakteristiku protokolu TCP/IP a generační rozdělení zařízení, které filtrují síťový provoz a to včetně problematiky frameworku Netfilter či nástroje iptables. Je rovněž uveden základní popis zvolené distribuce operačního systému Linux a virtualizačního nástroje, na kterém budou realizovány veškeré související nastavení a testy. Popis smyslu etického hackingu a výběr penetračního programu, který poslouží k otestování zvolené bezpečnostní politiky, je rovněž součástí této kapitoly.

Druhá část práce popisuje základní konfiguraci bezpečnostní politiky pomocí nástroje iptables, včetně charakteristiky a smyslu použití jeho jednotlivých částí. Zmiňuje použití nástroje Cacti, který byl zvolen za účelem monitorování vytvořené sítě. V této části bude rovněž provedeno testování zvolené bezpečnostní politiky a to včetně popisu souvisejících výstupů.

Závěr práce, tedy třetí, stěžejní část se zabývá charakteristikou a dělením síťových průniků. Popisem tří typů vybraných DoS útoků, které budou použity proti nezabezpečené a zabezpečené síti. Přibližuje tak práci s nástrojem iptables, vyhodnocování zaznamenaných grafů vzniklých průběhů a souvisejícím návrhem protiopatřením za účelem odepření nežádoucího průniku do sítě.

Cílem diplomové práce je nastavení bezpečnostní politiky paketového filtru s pomocí iptables a jeho využití pro maximalizaci ochrany sítě před síťovými průniky ve virtualizovaném Linuxovém prostředí.

1 Zabezpečení síťové komunikace

Počítačová síť je tvořena souborem hostitelů, kteří mezi sebou vzájemně komunikují. Tato komunikace je založena na speciálním jazyku, který se nazývá protokol. Protokoly jsou základem pro komunikaci v moderních datových sítích postavených na sadě TCP/IP. Abychom lépe porozuměli problematice filtrování datových spojení za účelem síťové bezpečnosti, je nezbytné porozumět protokolům, které tyto spojení uskutečňují [1].

Protokol TCP/IP a jeho použití

V současnosti komunikace v počítačové sítí probíhá pomoci rodiny protokolu TCP/IP. Místo přenášení celých souborů dohromady jsou data rozdělena do malých jednotek, tzv. paketů, které cílový hostitel po přijetí opět spojí. Model TCP/IP je složen z několika vrstev, a to konkrétně čtyř [2]:

- Vrstva síťového rozhraní
- Síťová vrstva
- Transportní vrstva
- Aplikační vrstva

Vrstva síťového rozhraní souvisí s přímým připojením k fyzickému médiu, po kterém jsou data přenášena. Mezi ty nejpodstatnější části síťové vrstvy patří bezesporu zdrojová a cílová IP adresa paketu. Transportní vrstva nabízí službu se spojením, nazývaný spolehlivý protokol (TCP) nebo bez spojení (UDP), takzvaně nespolehlivý protokol. Aplikační vrstva je nejvyšší vrstvou síťové architektury, sloužící k přenosu dat služeb. Některé aplikační protokoly vyžadují protokol TCP (např. FTP, HTTPS, TELNET apod.), jiné protokol UDP (např. SNMP, BOOTP, TFTP) [3]. Detailní popis je uveden v literatuře [3] [17] [18].

Virtualizace zvolené distribuce Linuxu

Pro realizaci své diplomové práce jsem si zvolil distribuci s názvem Debian 8 (Jessie) s prostředím GNOME. Debian je jednou z nejpoužívanějších distribucí GNU/Linuxu [5]. Jak již sám název tohoto odstavce napovídá, veškeré instalace, nastavování, testování a simulace síťových průniků budou realizovaný za pomocí virtualizace.

Pojem virtualizace, ve spojitosti s osobními počítači, nejčastěji vyjadřuje jakousi abstrakci uživatelského prostředí od fyzického hardware. Tedy technologii, kdy jsme schopni zdroje jednoho fyzického systému poskytnout několika izolovaným a na sobě nezávislým virtuálním prostředím, takzvaně strojům. Každý virtuální stroj tedy používá své vlastní virtuální prostředky, jako jsou síťové karty, souborové systémy, procesory, paměť a další. Tyto virtuální systémové zdroje používají tedy stejné sdílené fyzické zdroje, avšak tyto skutečné zdroje jsou virtuálním strojům skryty [5] [11].

Zvolil jsem nekomerční opensource nástroj pro virtualizaci s názvem VirtualBox. Při vytváření diplomové práce jsem musel vycházet ze skutečností, týkající se především reálné možnosti testovaní. Při použití virtualizace mohu vytvářet libovolné počty instancí a provozovat je virtuálně na

jednom fyzickém hardwaru. Díky tomu dojde ke značnému snížení nákladů, ať už finančních nebo spotřeby elektrické energie, neboť tak není potřeba pořizovat fyzické stanice pro testování. Mezi nejdůležitější argument, proč použít nástroj pro virtualizaci operačního systému Linux, považuji nepopíratelně právní a etickou odpovědnost, která souvisí s jakýmkoliv testováním síťových útoku ve veřejné síti. Díky této volbě bude afektován pouze stroj, na kterém budou prováděny simulace a nikdo jiný tak nebude nijak omezen či ohrožen ztrátou dat.

1.1 Filtrování síťového provozu

Filtrování IP provozu je jednoduchý mechanismus, který rozhoduje o tom, jaké pakety se mají předávat dál, zahodit nebo odmítnout. Zahozením je myšleno, že je paket vymazán a zcela ignorován jako by nikdy nebyl vůbec přijat. Odmítnutím je myšleno, že firewall odesílá ICMP zprávu příjemci oznamující důvod proč byl paket odmítnut. Můžeme použít mnoho různých druhů kritérií a určit, které pakety chceme filtrovat. Některé příklady z nich jsou [4] [17]:

- typ protokolu: TCP, UDP, ICMP, atd,
- číslo portu (pro TCP/UDP),
- typ paketu: SYN/ACK, data, ICMP Echo Request, atd.,
- zdrojová adresa paketu: odkud pochází.

Zařízení, které filtrují síťový provoz, slouží k řízení a zabezpečení provozu mezi sítěmi s různou úrovní důvěryhodnosti nebo zabezpečení. Dá se tedy zjednodušeně říct, že jsou jakýmsi kontrolním bodem, který je nadefinován pravidly a vytváří tak komplexní bezpečnostní politiku. Tyto zařízení prošly během let vývoje mnohými vylepšeními, nicméně jejich typové rozlišení je charakterizováno v následujících generacích.

První generace: nestavový paketový filtr

Paketový filtr (nebo také firewall) je v podstatě program, který je nejstarší a nejjednodušší formu firewallu. Mezi nejznámější paketové filtry patří ty, které pracují na síťové vrstvě (IP) a obsahují sadu pravidel obsahujících zdrojové a cílové adresy i zdrojová a cílová čísla transportních vrstev (TCP, UDP a podobně). Kontrola se provádí na třetí a čtvrté vrstvě modelu síťové komunikace OSI. Rozlišení, která komunikace bude povolena či zakázána, je řízeno bezpečnostní politikou. Tato politika je vložena do konfigurace firewallu a pro každý požadavek na průchod jsou aplikována pravidla bezpečnostní politiky, podle nichž paketový filtr rozhodne, jak s komunikací naloží [6].

Jak jsem již zmínil, filtrace síťového provozu probíhá na základě analýzy informací obsažených v hlavičce jednotlivého paketu, ale i validací například správné velikosti paketu nebo zda není paket úmyslně nebo neúmyslně poškozen. Paketový filtr je schopen kromě samotných informací obsažených v paketu brát v úvahu také porty, konkrétně pak port, na kterém paket do zařízení přišel a na kterém by měl odejít. Nestavový paketový filtr je tedy schopen následujících akci:

- poslat paket směrem k cíli,
- zahodit paket, aniž by byl odesílatel o této skutečnosti informován,

- odmítnout paket, ale odesílatele o této skutečnosti informovat,
- zapsat si informace o paketu do vlastního logu,
- vyrozumět administrátora, že se při filtraci byl objeven konkrétní paket,
- poslat paket jinému cíli, než pro koho byl původně určen,
- pozměnit filtrovací pravidla na základě konkrétního obdrženého paketu.

Výhodou tohoto řešení je vysoká rychlost zpracování, proto se ještě i dnes používají na místech, kde není potřebná důkladnější analýza procházejících dat. Nevýhodou je nízká úroveň kontroly procházejících spojení, která zejména u složitějších protokolů (např. FTP, video/audio streaming, apod.) ale mnohdy pro umožnění takového spojení vyžaduje otevřít i porty a směry spojení, které mohou být využity jinými protokoly, než bylo zamyšleno [6]. Mezi typické představitele nestavových paketových filtrů patří ACL (Access list) ve verzích operačního systému IOS od společnosti Cisco, nebo v linuxovém jádře jako ipchains.

Druhá generace: stavový paketový filtr

Stavové paketové filtry využívají podobného principu filtrování síťového provozu, jako nestavové paketové filtry s tím rozdílem, že navíc zahrnují možnost zapamatování si údajů o probíhajících spojeních. To tedy znamená, že provoz, který již jednou splnil požadavky nastavené bezpečností politiky, není po přijetí znova kontrolován. Tato vlastnost v sobě zahrnuje dvě výhody. První výhodou je urychlení zpracování paketů již povolených spojení a druhá výhoda spočívá v nastavení směru navázání spojení v samotných pravidlech. Stavový paketový filtr bude samostatně schopen povolit i pakety s odpovědí a u známých protokolů i další spojení, která daný protokol využívá [6].

Výhoda stavového filtrování je bezesporu ta, že není potřeba kontrolovat celé spojení, ale jen jeho část. Filtr navíc dokáže filtrovat na základě obsahu paketu a ne pouze jejich hlaviček. Proces filtrace paketů není nijak obtížný a hlavně zařízení zbytečně nezatěžuje, takže nijak neomezují síťový provoz a uživatelé o nich vlastně ani neví. V neposlední řadě disponuje vysokou rychlostí a dostatečnou úrovní zabezpečení v praxi, ale disponuje rovněž snadnou konfigurací. Snadná konfigurace filtru má spojitost s nižší pravděpodobností chybného nastavení pravidel. Nevýhodou je nižší úroveň zabezpečení v porovnání s moderními aplikačními filtry.

Myslím si, že klíčová výhoda stavového paketového filtru spočívá v jeho efektivitě, kdy strategickým umístěním stavové filtrace můžeme docílit vysoké bezpečnosti chráněné sítě. V Linuxu se můžeme s tímto tipem filtrování setkat u iptables. Právě zmiňovanou problematikou iptables ve spojitosti s realizací stavového paketového filtru, se v rámci této práce budu hlouběji zabývat.

Třetí generace: aplikační brány

Ve srovnání s nestavovým nebo stavovým paketovým filtrem jsou aplikační brány zcela odlišným systémem. Spojení zde probíhá přes takzvaného prostředníka, který zastupuje funkci Proxy brány. Tato metoda znamená, že jsou data kontrolována na aplikační vrstvě zmiňovanou Proxy bránou a následně předána dále v pořadí. Podstatnou skutečností je, že cílový prvek má informaci pouze

ohledně IP dresy Proxy serveru, přes který je komunikace řízena. Vše je realizováno formou spojení pomocí klienta, který se připojí na aplikační bránu (proxy bránu) která příchozí spojení vyhodnotí a zpracuje. Pokud je úspěšně vyhodnocena, vytvoří nové spojení k destinaci dle požadavku klienta. Informace které brána obdrží od destinace, poté v původním spojení předá klientovi [6]. Mezi výhody použití aplikačních bran patří bezesporu vysoká úroveň zabezpečení. Nevýhodou je ale vysoká náročnost na použitý hardware z důvodu vysokého požadavku na množství spojení a následného vyhodnocení. Nedosahuje se ani rychlosti kontroly jako na stavovém či nestavovém paketovém filtru.

1.2 Druhy linuxových firewallů

Existuje značné množství linuxových firewallu, které jsou zdarma k dispozici a umožnují tak administrátorům vytvořit bezpečnostní prvek v síti. Mezi ty nejznámější budu jmenovat například ClearOS, IPCop, OPNsence, IPFire či pfSence. Existuje i řada projektů, které si kladou za cíl stavbu firewallu zjednodušit, tedy být co nejvíce uživatelsky přívětivý a zaměřovat se tak na běžné uživatele. Příkladem popisované kategorie je nástroj Firestarter. Jiné typy nástrojů se pokouší usnadnit vytváření pravidel, ať již snahou o usnadnění návrhu komplikovaného firewallu (fwbuilder), nebo značným zjednodušením syntaxe (ufw, FireHOL, atd.) [15]. Nicméně mne nejvíce zaujalo použití bezpečnostního modulu iptables, který je součástí Netfilter. Netfilter nabízí mnoho možností při filtrování paketů nebo překladu síťových adres. Stavový paketový filtr, který budu v rámci své diplomové práce sestavovat, bude tak založen na konceptu použití iptables.

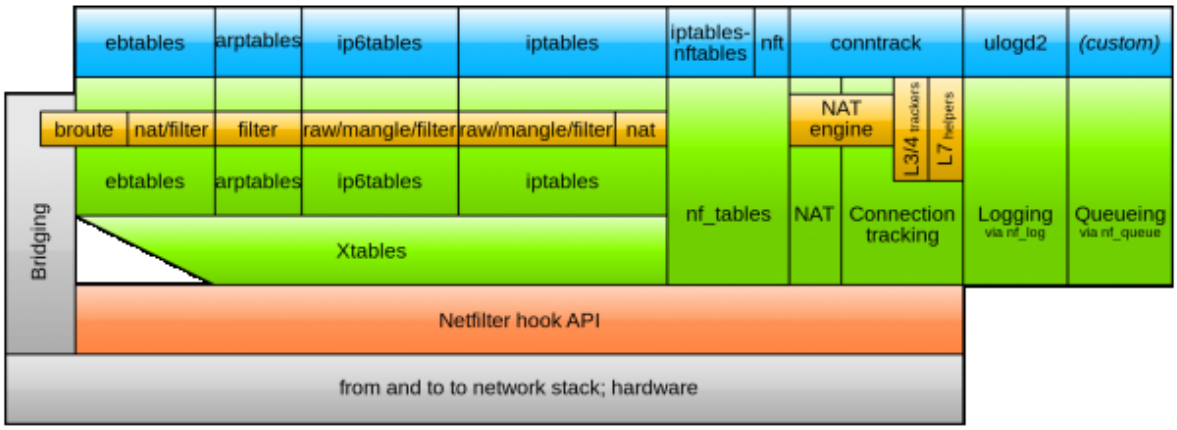
1.3 Netfilter

Netfilter představuje framework (rámec) pro práci s prostředky síťové komunikace na platformě Linux. Nabízí základní infrastrukturu určenou pro komunikaci kernelu s uživatelským prostředím, která je využita k předávání parametrů do určených modulů a zpětnému získávání stavových informací. Řeší jak komplexnost, tak i nedokonalost starších řešení implementací pomocí rámců jádra Kernelu. Díky tomu došlo k zjednodušení způsobu, jakým jsou pakety zpracovány a poskytuje se tím tak i možnost rozšíření schopností, aniž by muselo dojít ke změně samotného jádra.

V současné době můžeme říct, že se v tomto případě jedná o nejrozšířenější projekt v této oblasti, což dokazuje i ten fakt, že je bezesporu standardní součástí jádra operačních systémů typu Linux. Díky tomu, že je v praxi hojně využíván, je do jisté míry zaručen i rozvoj v případě nových verzí současně používaných protokolů. Nicméně Netfilter nefiltruje samotný síťový provoz, jen umožňuje využít funkce uvnitř jádra Kernel tak, aby byl provoz filtrován patřičnými moduly. Chceme-li tedy vytvořit linuxový filtr, je nezbytné využit službu modulů jádra iptables, ip6tables, arptables nebo ebtables. Tyto moduly poskytují systém pro definování pravidel firewallu pro filtrování nebo transformování paketu s využitím příkazové řádky potřebné pro komunikaci s jádrem Kernelu. Výhodou Netfilter je zpětná podpora a kompatibilita s oběma historicky staršími verzemi linuxových filtrů (ipfwadm a ipchains) ve spojitosti s novým nástrojem iptables [5] [17]. Z hlediska filtrování provozu se budu zabývat vlastnostmi a použitím modulu iptables, jelikož mi jeho funkce v této diplomové práci plně dostačují.

1.3.1 Architektura

Netfilter jako takový, je tvořen přípojnými místy, která jsou umístěna v protokolovém zásobníku. Tyto přípojná místa dokážou vyvolat obslužné metody. Rozmístění jednotlivých bodů je zvoleno pečlivě a každé umístění má své specifické odůvodnění. V místě přípojných bodů se pak ze síťového kódu jádra volají metody registrovaných modulů. Každá vyzvaná funkce registrovaná ke konkrétnímu jednomu z přípojných bodů má k dispozici paket, se kterým může dle svého uvážení libovolně nakládat. Každá funkce modulu má svou prioritu, podle které jsou jednotlivé funkce v rámci jednotlivých modulů volány [17] [5]. Strukturu celého frameworku lze vidět na obrázku 1.1.



Obrázek 1.1: Komponenty Netfilter
[Zdroj:http://www.systutorials.com/wp/wpcontent/plugins/blogtext/api/thumbnail/do.php?id=55a0a431747156eb785b71c11fa1e3b7e64bfddf_630x0_resize_if_larger]

Modul může libovolný paket určovat následovně:

- zajistit pokračování procházení paketu beze změny (CONTINUE),
- poslat paket z jádra do uživatelského prostoru a zařadit jej do předem určené fronty (QUEUE),
- zahodit paket a ukončit jeho procházení zásobníkem (DROP),
- zavolat opětovně metodu přípojného místa (REPEAT),
- ukončit procházení zásobníkem, protože se modul dostal „přes“ samotný paket (STOLEN).

Musíme mít na mysli, že pokud paket zařadíme do uživatelské fronty, je nezbytné jej v do určitého časového intervalu vyvolat zpátky do jádra či jinam, aby nedošlo ke ztrátě samotného paketu nebo případnému zpoždění komunikace [5] [18]. Součástí rámce Netfilter je tedy již zmíněný modul iptables. Ten je tvořen takzvanými tabulkami (tables), kterými procházejí samotné pakety. Jedná se o čtyři základní tabulky, které jsou NAT, MANGLE, FILTER a RAW.

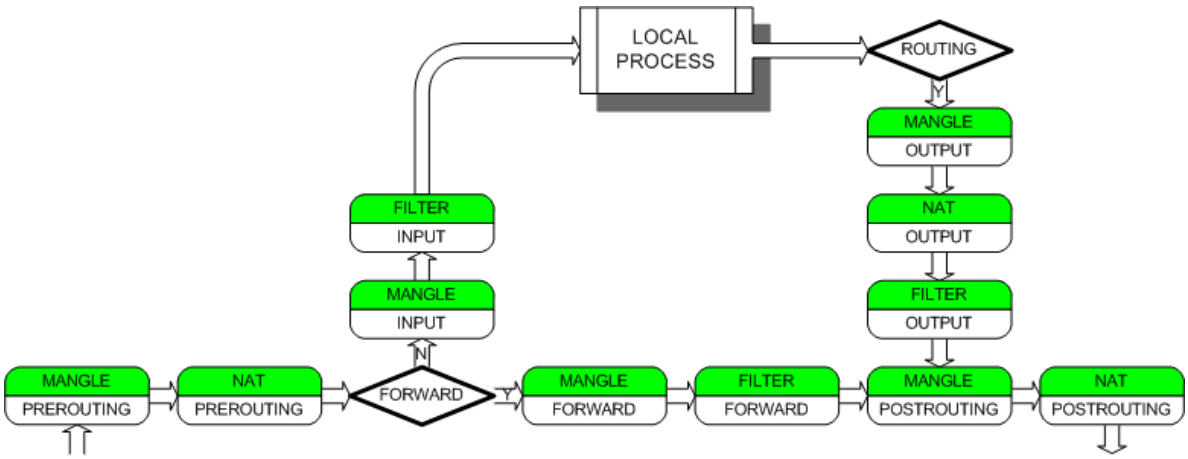
NAT tabulka zajišťuje překlad síťových adres. Je připojena k řetězcům PREROUTING, POSTROUTING a OUTPUT. **MANGLE** tabulka se používá k určitým modifikacím informací v rámci paketu. Tato tabulka je připojena ke všem přípojným řetězcům. **FILTER** tabulka je pro nás v případě filtrování nejvíce podstatná a výchozí. Tato tabulka neumožnuje jednotlivé pakety nijak měnit, ale pouze je filtrovat. Tabulka je připojena k bodům INPUT, OUTPUT a FORWARD. Význam

těchto bodů bude popsán dále v této kapitole. **RAW** tabulka umožňuje nastavit značku na pakety, které by neměly být řešeny prostřednictvím sledování stavu spojení. Souvisí pouze s řetězci PREROUTING a OUTPUT. Protože to jsou jediná místa, kde je možné zacházet s pakety před tím, než bude jejich stav spojení sledován [5].

V rámci této práce budu pracovat pouze se dvěma popsanými tabulkami **FILTER** a **NAT**. Tabulka MANGLE použita nebude, jelikož obsahuje sady pravidel pro úpravy hlavičky paketů, či například manipulaci s TTL nebo TOS, které můj koncept práce nezahrnuje. I tabulku RAW nebudu používat, protože funkcionalita sledování stavu spojení je v případě sestavování stavového paketového filtru nutnou vlastností.

1.3.2 Filtrování paketu

Pokusím se nastínit příklad průchodu paketu přípojnými body na následujícím obrázku 1.2, pro lepší objasnění problematiky. Tabulka MANGLE bude v popisu průchodu paketu vyňata z výše popsaných důvodu nevyužití při realizaci této práce.



Obrázek 1.2: Průchod paketu přípojnými body Netfilter

[Zdroj: http://ebtables.netfilter.org/br_fw_ia/bridge3b.png]

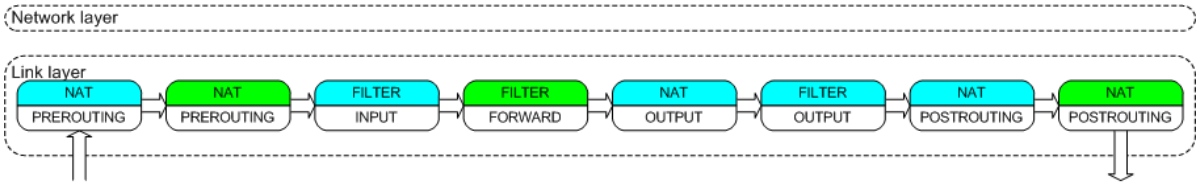
Po přijetí paketu do operačního systému se jako první uskuteční prvotní kontroly (např. kontrolní součet). Poté je vyvolán první bod, NAT (PREROUTING). Dále se rozhoduje, zda je paket určen danému zařízení nebo se bude směrovat na jiný port zařízení. Podle toho jsou vyvolány další přípojné body FILTER (INPUT) nebo FILTER (FORWARD.) V případě směrovaného paketu pak tento paket projde k závěrečnému bodu NAT (POSTROUTING) a je následně odeslán [5].

Odchozí IP pakety, které jsou vytvořeny samotným zařízením, jsou po předání do protokolového zásobníku nejprve odeslány k přípojnému bodu FILTER (OUTPUT), poté se pro ně spustí směrovací kód a dále jsou předány závěrečnému přípojnému bodu NAT (POSTROUTING). Ve skutečnosti je směrovací kód spuštěn před odchozím bodem FILTER (OUTPUT), aby byly vypočteny zdrojová adresa a některé další možnosti IP [5].

I přesto, že je framework Netfiler spíše zaměřen na vrstvu síťovou se souvisejícím IP protokolem a modulem iptables, je vhodné porozumět i části problematiky týkající se použití Netfilter v rámci vrstvy linkové. Tedy ve smyslu použití pro případ, kdy je na daném zařízení s operačním systémem Linux nakonfigurován logický most (bridge). V tomto případě pakety (ve formě rámců) prochází zásobníkem v kódu linkové vrstvy, která má nadefinované a připravené přípojné body. V opačném případě rámce putují rovnou do vrstvy síťové jako pakety. Detailní popis související problematiky je uveden v literatuře [6] [18].

1.3.3 Netfilter na síťové vrstvě

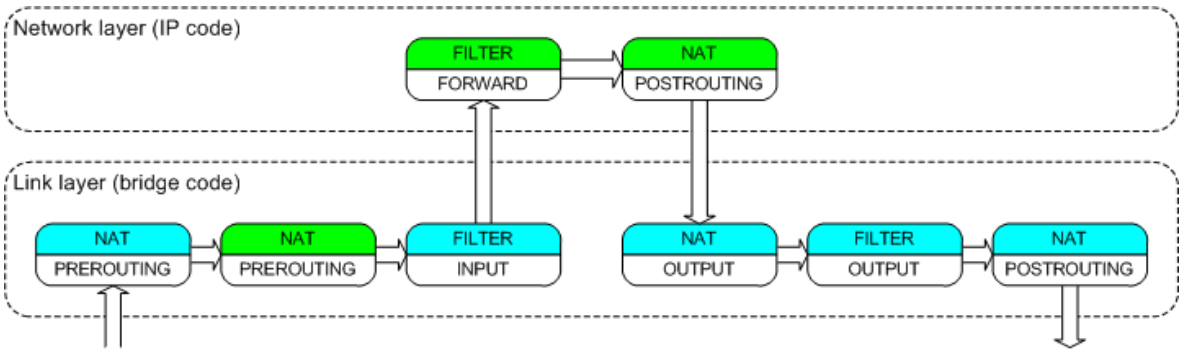
Linkovou a síťovou vrstvu lze samozřejmě zkombinovat, nicméně průchod paketu skrze jejich jednotlivé přípojné body a průchod jednotlivými tabulkami na jednotlivých vrstvách není vždy totožný. Významnou roli zde zastává vlastnost samostatného paketu zejména pak na příchozí a odchozí port. Mohou nastat tři případy a s nimi spojené putování paketů v rámci systému. Tyto tři případy se pokusím rozvést a objasnit. Pokud nastane situace, že jsou pakety předávány z jednoho portu mostu na druhý, nedostanou se tedy vůbec na síťovou vrstvu. V tomto případě však Netfiler volí možnost odesílat pakety skrze tabulky iptables [6]. Obrázek 1.3 popisuje tuto situaci.



Obrázek 1.3: Řetězec průchodu paketu v iptables

[Zdroj: http://ebtables.netfilter.org/br_fw_ia/bridge6a.png]

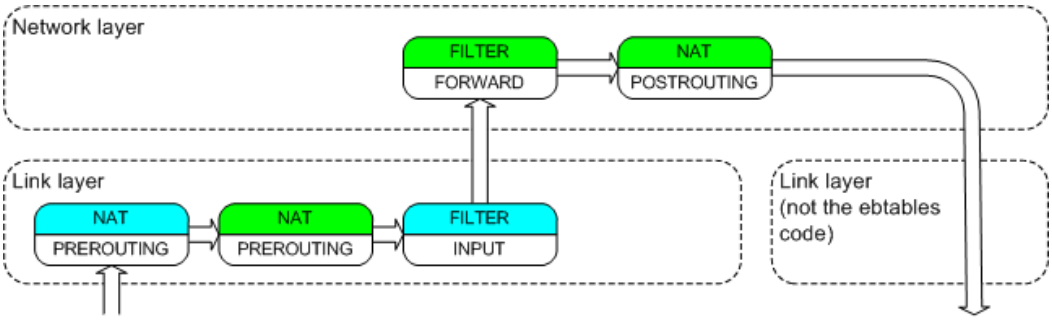
V případě, že je v rámci Netfilter použito zařízení v kombinaci most a směrovač, průchod paketu je odlišný. Obrázek 1.4 znázorňuje situaci, ve které je rámec určen na linkové vrstvě pro samotné zařízení, nicméně bude následně přeposlán síťovou vrstvou, kdy opouští zařízení jedním z portů mostu [6].



Obrázek 1.4: Směrování paketu na port mostu

[Zdroj: http://ebtables.netfilter.org/br_fw_ia/bridge3c.png]

Pokud však nastane situace, že se na síťové vrstvě paket rovnou přepošle na port, který už není součástí mostu, není paket nucen uskutečňovat cestu skrze tabulky linkové vrstvy [6]. Obrázek 1.5 znázorňuje tento případ.



Obrázek 1.5: Směrování paketu na port směrovače

[Zdroj: http://ebtables.netfilter.org/br_fw_ia/bridge3d.png]

Z pohledu filtrování paketů, kterým se tato diplomová práce bude ubírat, jsou především důležité tabulka FILTER a její řetězce. Nicméně po zamyšlení se na tím, kudy bude jednotlivý paket opravdu procházet a jak bude následně zpracován, spočívá v rozhodovacím režimu samotného mostu nebo směrovače. Je však odvislé od samotné konfigurace zařízení a portu, na jakém je paket přijat. V tom případě, že je paket rovnou při přijetí poslán do směrovací části, automaticky se zde přichází o možnost filtrování na základě informací z hlavičky linkové vrstvy. Stejné chování pak lze pozorovat na straně odesílání paketů, kde možnost filtrace určuje linková vrstva a rovněž závisí na konfiguraci portu, kterým paket zařízení opouští.

1.3.4 Stavy spojení

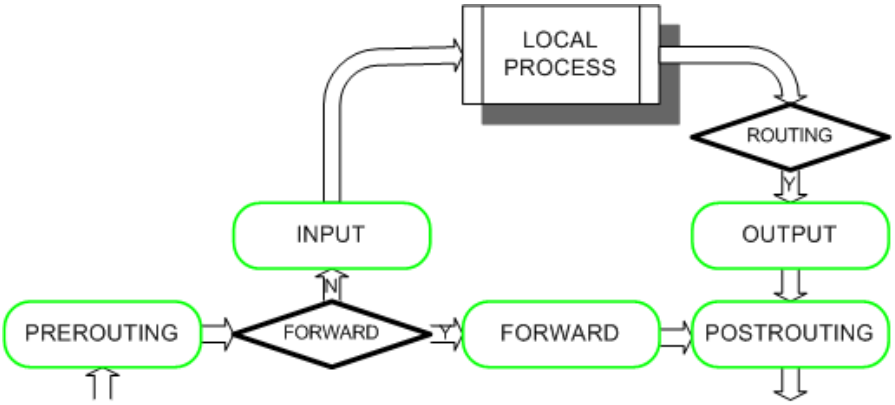
Stavy spojení jsou důležitou vlastností samotného Netfilter, díky kterým stavový paketový filtr poskytuje pokročilé mechanismy umožňující administrátorům a bezpečnostním expertům mít přehled o všech sítových spojeních a samotných relacích. Dny, kdy byla filtrovací politika založena unikátně na informacích z hlavičky paketu, jako to IP adresa zdroje, IP adresa cíle a číslo portu je již nedostačující. V průběhu let se tento přístup prokázal jako nedostatečná ochrana proti síťovým útokům typu DoS [17].

Modul sledování spojení označuje každý paket jedním ze čtyř možných stavů a to NEW, ESTABLISHED, RELATED, INVALID. Jednotlivé stavy spojení se vytvářejí z příznaků TCP protokolu. První paket, který paketový filtr rozezná je určen stavem NEW. V případě, že paketový filtr obdrží TCP paket s příznakem SYN ACK tak i přesto, že se teprve jedná o druhou fázi navazování spojení pomocí trojcestného handshakingu a dochází teprve k potvrzování nikoliv už navázání TCP spojení, paket má přiřazen stav ESTABLISHED. Paketový filtr tedy odpovídá na tento paket stavem ESTABLISHED a případné chybové hlášení protokolu ICMP vztahující se k tomuto spojení pak bude ve stavu RELATED. Paket, který se nevztahuje k žádnému souvisejícímu spojení, je brán jako paket neplatný a má proto určení INVALID [5].

Popsaný mechanizmus stavů spojení nefiltruje pakety, ale pouze je sleduje. V podstatě ukládá informace o stavu spojení ve struktuře paměti, která obsahuje IP adresu zdroje a cíle, číslo portu, typy protokolů, stavy a timeout. Díky tohoto rozšíření můžeme definovat více inteligentní filtrovací politiku [17] [18].

1.4 Iptables

Jak již bylo zmíněno, iptables je obecná tabulková struktura, určená pro definování sady pravidel za účelem síťové filtrace vycházející z Netfilter. Umožnuje uživateli ovládat vysoký stupeň kontroly IP paketů pomocí politiky, která ve svém jednouchém základu rozhoduje, zda bude paketu přenos povolen či zamezen. Politika iptables je postavena jako uspořádané nastavení pravidel, které popisují kernelu, jak má být s každým paketem zacházeno.



Obrázek 1.6: Tok paketu v iptables

[Zdroj: http://ebtables.netfilter.org/br_fw_ia/bridge3a.png]

Každé pravidlo se aplikuje do řetězce v tabulce (obrázek 1.6). Řetězec je množina pravidel, které porovnávají pakety se společnou charakteristikou, jako je například cílové směrování [5].

Tabulky (Tables)

Jak jsem již zmínil v podkapitole zabývající se problematikou Netfilter, budu ve své práci využívat pouze tabulky **FILTER** a **NAT**, které nyní blíže popíši:

- FILTER — umožňuje zachytit a filtrovat jakýkoliv paket a provést patřičnou akci s ohledem na jeho obsah. Podstatnou vlastností je, že každý paket prochází filtrační tabulkou právě jednou.

- NAT — tato tabulka by měla být použita pouze pro překlad síťových adres paketů. Pouze první paket v pořadí bude vyhodnocen touto tabulkou, zbylé související pakety budou automaticky spadat pod stejná opatření, jako tomu bylo v případě prvního paketu.

Řetězce (Chains)

Každá tabulka má svou vlastní sadu zabudovaných řetězců, které mohou být rovněž samotným uživatelem. Základní tabulka FILTER obsahuje tři základní směry, kudy může paket procházet. Jedná se o tyto řetězce:

- INPUT určuje pravidla pro příchozí pakety,
- OUTPUT určuje pravidla pro odchozí pakety,
- FORWARD určuje pravidla pro příchozí pakety, které jsou následně směrovány v samotném systému Linux, nebo pro směrování paketů mezi dvěma různými sítěmi.

Tabulka NAT obsahuje tři řetězce, které jsou použity k modifikaci adres paketů. Je to již popsaný řetězec OUTPUT a tyto dva zbylé:

- PREROUTING je jako první v pořadí, tedy než dojde k jakémukoliv směrování paketu. V tomto řetězci se provádí DNAT.

- POSTROUTING obsahuje pravidla použitá v procesu směrování. V tomto řetězci se provádí SNAT [5].

Cíle (Targets)

Cíle představuje akci, která se provede s paketem po nalezení shody. Nejpodstatnějšími použitými akcemi jsou:

- ACCEPT paket je přijat bez ohledu na všechny následující pravidla,
- DROP paket je okamžitě zahozen,
- LOG zaznamenává informace o paketu do syslogu,
- REJECT zahazuje pakety a současně odesílá paket s informací,
- RETURN pokračuje ve zpracování paketu uvnitř volaného řetězce.

Mezi často používanou akci patří LOG, který ale není ukončujícím pravidlem. Pokud paket splňuje pravidlo LOG, jádro pokračuje dalšími pravidly, jenž tento paket splňuje [5].

Překlad IP adres

Překlad IP adres neboli NAT (Network Address Translation). Jedná se o nedílnou součást filtrační oblasti, která je použita jednak z důvodu nedostatku veřejných IP adres, ale taky z důvodu bezpečnostního. Princip je zcela jednoduchý, pokud paket při průchodu vyhoví zadanému pravidlu, tak mu je podle určeného vzoru změněna adresa jeho odesílatele resp. příjemce podle určení. Na základě této skutečnosti poté dělíme NAT na SNAT (zdrojový překlad síťových adres) nebo překlad adres příjemce DNAT (cílový překlad síťových adres). SNAT provádí maskování IP adresy směrovaných paketů na adresu vnějšího rozhraní paketového filtru. DNAT využíváme, máme-li potřebu změnit IP adresu adresáta paketu [5].

Jistou alternativou SNAT je použití maškarády. Ta umožňuje zařízením, které mají ve vnitřní síti přiděleny privátní IP adresy, zpřístupnit služby v Internetu (například přístup na WWW servery a podobně). Používá se v případě, že nemáme dostatek veřejných IP adres pro všechny vnitřní stanice v lokální síti. Stanice v takovém případě mají přidělenu privátní IP adresu, ale s pomocí maškarády používají pro přístup do Internetu veřejnou IP adresu, která je přidělena maškarádujícímu zařízení. Toto zařízení tedy přepisuje zpáteční IP adresu na svoji vnější IP adresu (či adresu svého vnějšího rozhraní) a při příchodu odpovědi přepíše cílovou IP adresu zpět na původní stroj z lokální sítě.

1.5 Etický hacking a penetrační nástroje

Vyhodnocovat úroveň zabezpečení počítačové sítě je možné pomocí penetračních testů, které spočívají v simulaci možných útoků na systém nejen uvnitř sítě. Výsledky testů by pak měly vést k případným nápravám bezpečnostních nedostatků. V souvislosti s penetračním testováním hovoříme o tzv. etickém hackingu, kde není účelem nikomu uškodit. Úkolem etického hackera je najít v testovaném systému zranitelnost a posoudit, jak by je dokázal nepřítel zneužít [14] [14]. Etický hacking lze charakterizovat jako:

- pochopení činností a nástrojů hackerů za účelem zlepšení bezpečnosti vlastních systémů,
- etický hacker rozumí slabým a zranitelným místům v systémech a při jejich ochraně využívá znalostí o chování útočníků.

Je doporučeno zkoušet hacking pouze a jen ve svém fyzickém testovacím prostředí. V případě použití alternativy virtualizace, zvolit jeden virtuální stroj v roli oběti a druhý v roli útočníka. Ideálně pak celou síť s dalšími virtuálními stroji. Každý nástroj je psaný pro nějaký konkrétní úkol, síťovou službu nebo aplikaci. Pokud nebude na cílovém virtuálním stroji běžící služba, která na síti poslouchá na specifické IP adrese a portu, náš testovací nástroj nemůže provádět žádné interakce. Pokud by to přeci jen někoho napadlo a začal by testovat na cizích systémech, tak nejspíše brzy skončí s omezením na osobní svobodě. Je třeba si uvědomit, že k porušení práva dochází už jakýmkoli neautorizovaným přístupem bez ohledu na vyspělost ochrany. I když úmysly testera mohou být čistě naučné, špatně zvolený nástroj nebo jeho použití (překlep, nepochopení syntaxe) může vážně poškodit běžící proces, vyčerpat zdroje na síti, disku nebo CPU. V takovém případě může pomoci jen zásah administrátora, restart procesu nebo celého serveru. Na produkčním systému začnou administrátoři hledat příčinu a auditní stopy, jak v logovacím souboru cílového serveru, tak na síťových prvních poskytovatele připojení. Smyslem produkce je, aby generovala zisk nebo nepřímo podporovala jiné procesy v organizaci. Ztráta způsobená výpadkem takového systému začíná v jednotkách minut a stupňuje se podle důležitosti procesu, který daný systém podporuje. Nejčernější scénář je obnova systému ze zálohy nebo dokonce selhání a výměna hardwaru, to už je pak záležitost hodin.

Jak již bylo napsáno v úvodu, smyslem penetračního testování je ověření úrovně zabezpečení aplikace, systému nebo sítě. Někdy se používá zkrácený výraz pen-test. Provádí se testováním, hledáním slabého místa, a následně pokusy proniknout do infrastruktury a pokud možno získat co nejvyšší oprávnění. Nakonec zbývá interpretace případných nedostatků. Testováno by mělo být vše,

u čeho hrozí riziko nežádoucího průniku do systému, odcizení dat nebo způsobení finanční škody. Nejčastěji po útoku vznikají škody typu:

- nedostupnost služby, tedy služba není schopna obsluhovat legitimní uživatele (DoS, DDoS útoky),
- neoprávněný přístup útočníka k systému (počítač, server, databáze), kde může číst či modifikovat data, měnit konfigurace,
- získání důvěrných informací, například získání přihlašovacích údajů, adres, informací o financích apod.

Pokud jsou nalezena nějaká slabá místa, může se naskytnout možnost pro napadení systému. Dnešní nástroje pro odhalování zranitelností pracují po zadání vstupních údajů a provedením požadovaného nastavení většinou automaticky. Výčet typických činností těchto nástrojů:

- procházení otevřených portů a služeb v celém bloku IP adres,
- zjištění typu operačního systému a aplikací, jejich verze, nainstalované záplaty,
- zjištění nastavení, zabezpečení, autentizace aplikací nebo služeb,
- některé dovedou i nízko úrovňové hádání hesel hrubou silou,
- poskytnutí informací o síťové službě a její verzi, operačním systému síťových prvků.

Výsledky testů odhalují základní bezpečnostní nedostatky systému a na testující osobě spočívá úkol vyhodnotit, které problémy představují riziko v kontextu testovaného prostředí. Může se stát, že nebezpečné chyby, označené automatickým softwarem, nemusí v daném prostředí představovat vážné reálné riziko. Naopak malá detekovaná chyba může vést k většímu útoku [14] [16].

Typy penetračních testů a jejich dělení

Testy je možné dělit podle různých hledisek. Obecně se penetrační testy často dělí na externí a interní. Externí testy jsou prováděny z vnější strany testované sítě a představují vnější hrozby (např. útok hackera z internetu). Interní testy jsou prováděny z vnitřní strany testované sítě, které napodobují potencionálního útočníka, který získal nějakým způsobem přístup do vnitřní sítě, nebo také neloajálního zaměstnance. Podle svého způsobu provedení se testy dělí na manuální a automatické. Manuální testy vykonává tester manuálně s možností vytvořit testy na míru pro specifické podmínky. Nevýhodou je, že jsou potřeba rozsáhlé znalosti testované oblasti a dovednosti vytvořit testovací proceduru. Další nevýhodou je časová náročnost. Automatizované testy jsou nástroje pro automatické testování. Vytvářejí je profesionálové v oboru a testerovi se stačí naučit s nástrojem pracovat a porozumět interpretaci výsledků. Výhodou je rychlost aplikace testu, nevýhodou může být nemožnost otestovat některé typy zranitelných míst.

Rozdělení penetračních testů se týká také účinnosti, jakou působí na zatíženou bezpečnostní politiku. Penetrační testy se dělí na destruktivní a nedestruktivní. Destruktivní testy prakticky ověřují objevené zranitelnosti, což může vést k selhání testovaného systému, například během testování systému pomocí napadení útokem DoS či DDoS. Za to nedestruktivní testy mohou odhalit chyby ještě před tím, než je použito razantní destruktivní testování. Dalším dělením může být množství informací,

které o zkoumaném systému útočník má. **Black box** testování lze označit jako více podobné útočníkovi, který má pouze minimální znalosti. **White box** lze přirovnat k útočníkovi, který například v určité organizaci dříve pracoval [14].

Zvažoval jsem, použitím kterého z dostupných nástrojů bych vhodně popsal problematiku týkající se penetračního testování. Některé nástroje jsou komerční, ale většina jich je zdarma, protože jsou často vyvíjeny hackerskými komunitami a sdíleny na internetu. Existuje také řada operačních systémů zaměřených na bezpečnostní testování. Typickým příkladem jsou různé distribuce Linuxu, které obsahují širokou škálu ověřených nástrojů různých vývojářů, vyvinutých třeba i pro jeden účel [14]. Zvolil jsem použití linuxové distribuce s názvem Kali linux, která svými vlastnosti ideálním způsobem odpovídá mým požadavkům.

Kali Linux

Kali Linux je bezplatná linuxová distribuce odvozená od distribuce Debian. Jedná se o pokročilou distribuci určenou k penetračnímu testování a bezpečnostním auditům. Rozdílem oproti ostatním linuxovým distribucím je využití super uživatele (root). Většina nástrojů totiž vyžaduje rootovská práva. Dalším rozdílem je výchozí zákaz většiny síťových služeb. Povolené a zakázané služby můžeme zobrazit/editovat v takzvaných **blacklistech** a **whitelistech** [15].

Využil jsem vzhledem ke zvoleným nástrojům při tvorbě této práce možnost stáhnout si připravený Kali Linux jako obraz pro virtuální stroj. Myslím si, že virtuální stroje jsou pro účely testování jako stvořené. Při výskytu jakékoliv námi vytvořené chyby při modifikaci nabízených nástrojů není nic jednoduššího, než si obnovit virtuální obraz systému a začít svou práci od samého začátku.

Vybrané nástroje v Kali Linux

K otestování a vyhodnocení zvolené bezpečnostní politiky paketového filtru jsem vybral nástroje **Nping**, **Nmap** a **Wireshark**. Tyto tři nástroje jsou v praxi díky svým vlastnostem velice používané a jejich postupný vývoj došel do fáze, kdy předčí mnohé konkurenty ve svém oboru. **Nmap** i **Nping** patří mezi jedny z nejpoužívanějších nástrojů určených pro skenování portů nebo k prověření bezpečnosti systému. Slouží jako základní prvek penetračního testování. Dokáží jednoduše prověřit funkčnost paketového filtru a zjistit informace o dostupných službách na spravovaném systému tak, jak by to dokázali případní útočníci. Nástroje mají i mnoho pokročilých funkcí, umí nejen detekovat běžící služby, ale i verzi operačního systému a dobu jeho spuštění. [14]. Jejich použití je snadné, využívají příkazového řádku. Nástroj **Wireshark** slouží k analýze síťových paketů. Jeho hlavní funkcí je záchyt paketů ze sítě, jejich dekódování a následná prezentace zahrnující výčet velmi podrobných informací, které lze z paketů získat. Jedná se o open source program, který nabízí základní funkce jako TCPdump, avšak s řadou vylepšení.

2 Návrh paketového filtru

V této části kapitoly se pokusím popsat praktické návrhy pro efektivní sestavení bezpečnostní politiky paketového filtru, které jsem použil. Nastavenou politiku rovněž otestuji zvolenými nástroji, abych se ujistil, že splňuje mnou určené požadavky. Na závěr kapitoly popíši zvolený monitorovací nástroj Cacti pro vytváření grafů síťového provozu, jenž napomáhá identifikovat a odhalit možné charakteristiky probíhajícího útoku.

2.1 Specifikace a parametry filtrovacího pravidla

Pro úspěšný návrh stavového paketového filtru je prvním krokem vytvoření filtrovacího pravidla. Toto pravidlo představuje jakousi množinu nastavení jednotlivých voleb v hlavičce paketu, vůči které se samotný paket porovnává. Výsledkem porovnání je shoda nebo neshoda. V případě shody pak pravidlo definuje, co se s daným paketem udělá.

Můžeme nadefinovat pravidlo, které bude aplikováno v případě přesné shody například se zdrojovou IP adresou. Tato možno je však poněkud zavádějící a do jisté míry i nepraktická, protože v případě požadavku na komunikaci z určitého segmentu sítě (síť se stejným prefixem adresy, tedy maskou sítě) bychom museli nadefinovat pro každé zařízení pravidlo zvlášť. Řešením je vytvoření jednoho pravidla, které obsahuje právě adresu sítě společně s maskou sítě, jako to IP adresu zdroje. Toto pravidlo se poté vztahuje na všechny zařízení daného segmentu.

Další možností při vytváření filtrovacího pravidla je negace hodnoty. Pro vysvětlení uvedu opět názorný příklad. Pokud budeme chtít povolit veškerou komunikaci kromě protokolu TCP, lze jednoduše povolit každý známý protokol zvlášť. Nicméně z praktického hlediska je jednodušší jedním pravidlem zakázat pakety nesoucí TCP protokol a jedním pravidlem povolit pakety, které nesou „něco jiného“, než TCP, tj. takové pakety, kde se informace v hlavičce indikující protokol nerovná protokolu TCP. Samozřejmě ve filtrovacím pravidle nemusíme hodnotu vůbec specifikovat. V tom případě bude tato hodnota při porovnání vynechána a nebude na ni brán zřetel. Jinak řečeno, pravidlo zafunguje pro jakoukoliv hodnotu daného parametru.

Jelikož navrhovaný paketový filtr má obsahovat univerzální nastavení své bezpečnostní politiky, měl by být tedy schopen filtrovat síťový provoz i na linkové vrstvě zároveň. Zvolil jsem tyto parametry, podle kterých lze pakety filtrovat:

- vstupní a výstupní porty (do / ze zařízení),
- na linkové vrstvě zdrojová a cílová hardwarová adresa,
- na linkové vrstvě typ použitého protokolu síťové vrstvy,
- na síťové vrstvě zdrojová a cílová hardwarová adresa (včetně masky sítě),
- na síťové vrstvě typ použitého protokolu vyšší transportní vrstvy.

Vstupní a výstupní porty není potřeba nějak více vysvětlovat. Jedná se o port, na kterém byl paket do zařízení přijat případně ze zařízení odeslán. Tento parametr se nevztahuje k linkové ani síťové vrstvě, avšak jedná se o důležitý parametr, který by neměl zůstat opomenut a pokusím se přiblížit jeho význam. V reálném prostředí probíhá filtrace v hraničních bodech počítačové sítě, které

mohou spojovat několik segmentů, z nichž každý je připojen na jeden či více portů. Jsem tak schopni do jisté míry říci, jaké adresy se mohou objevit na určitých segmentech a tím i konkrétních portech. V případě, že se objeví na některém portu námi neočekávaná adresa, může být velmi často považována jako indikace pokusu o útok. Zdrojové a cílové hardwarové adresy linkové i síťové vrstvy jsou také velmi jasným parametrem. V obou případech, jak již bylo popsáno v textu výše, lze zadat rozsah adres s maskou sítě.

Typ protokolu přenášeného linkovou vrstvou neboli typ protokolu síťové vrstvy představuje parametr s omezeným, avšak velkým, výběrem možností. Uživateli tak není dovoleno specifikovat parametr napsáním libovolné hodnoty. Zde jsem uvedl některé nejčastější možnosti tohoto parametru:

- IP protokol verze 4 (IPv6),
- IP protokol verze 6 (IPv4),
- ARP protokol (ARP).

Mezi další, dle mého názoru velice podstatné, parametry pravidla patří název a popis. Název by měl být výstižný a krátký, jelikož slouží pro jednoznačnou identifikaci v seznamu pravidel při správě a konfiguraci firewallu. Naopak popis může být již delší a konkrétnější než samotný název, protože dává administrátorovi (či více administrátorům) informaci například o dočasném použití, odkazu na jiné související pravidlo či důvodu samotné implementace.

Je důležité si uvědomit, že všechny parametry pravidla platí současně v jeho definici. Mluvíme zde tedy o logickém součinu (AND) mezi nadefinovanými parametry. Pouze v případě, že paket vyhovuje všem parametrům najednou, je jeho další zpracování závislé na definované akci onoho pravidla. V opačném případě probíhá porovnání s dalším pravidlem v řadě. Pokud tedy definujeme pravidlo pro filtraci na základě hlaviček protokolů použitých na linkové a síťové vrstvě zároveň, musíme brát v potaz tu skutečnost, že daný paket/rámec musí splňovat požadavky definované v pravidle zároveň. Nelze tímto pravidlem ovlivnit pakety, které ačkoliv mají například stejnou hlavičku protokolu síťové vrstvy (standardně IP) a nesplňují už požadavky na linkové vrstvě. Ovlivnění takových paketů je považováno za chybnou interpretaci.

2.2 Základní sestavení bezpečnostní politiky paketového filtru

Většina linuxových distribucí má již nainstalován nástroj iptables a je tedy ihned připraven k využití. Tato skutečnost platí i pro mnou zvolenou linuxovou distribuci Debian, kterou budu ve své diplomové práci používat. Avšak pro kontrolu a ujištění, že je systém připraven na realizaci mého paketového filtru jsem tyto skutečnosti ověřil:

1
```
root@PaketovyFiltr:/home/michal# iptables -V
iptables v1.4.21
```

Pomocí příkazu na zobrazení verze iptables (**1**) jsem si ověřil, že systém pracuje s aktuální možnou verzi, která je v současnosti k dispozici.

2

```
root@PaketovyFiltr:/home/michal# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

Rovněž nastavení pravidel iptables (**2**) potvrdilo, že aktuálně není použito žádné pravidlo pro filtraci síťového provozu, tedy provoz je povolen ve všech směrech. Nyní tak máme ověřeno, že pracujeme s funkčním systémem Linux a to včetně jeho nástroje iptables.

Požadavky na politiku

Nyní si určím požadavky pro efektivní nastavení paketového filtru. Jelikož jsem se rozhodl při tvorbě své diplomové práce využít již dříve popsanou virtualizaci, díky které mohu vytvářet virtualizované prostředí mezi platformou počítače a operačním systémem, je nezbytné si stanovit počet zařízení, se kterými budu pracovat. Omezujícím faktem, kterému musím v této situaci čelit, je hardwarové vybavení fyzického počítače. Jsem tak nucen omezit se pouze na vytvoření třech instancí, které se budou týkat stanice s paketovým filtrem, serveru jako to cíle útoku a útočníka, jenž bude iniciovat síťové útoky pomocí specifických nástrojů.

Uvedu následující modelovou situaci, která odpovídá testování v laboratorním prostředí, v němž budu pracovat a poslouží k objasnění problematiky práce s iptables. Tedy, server musí být dostupný z externí sítě (internet) pouze pro určité služby a zároveň by měl být povolen přístup určitému provozu, který se bude iniciovat ze serveru přes firewall směrem do externí sítě internet, a to:

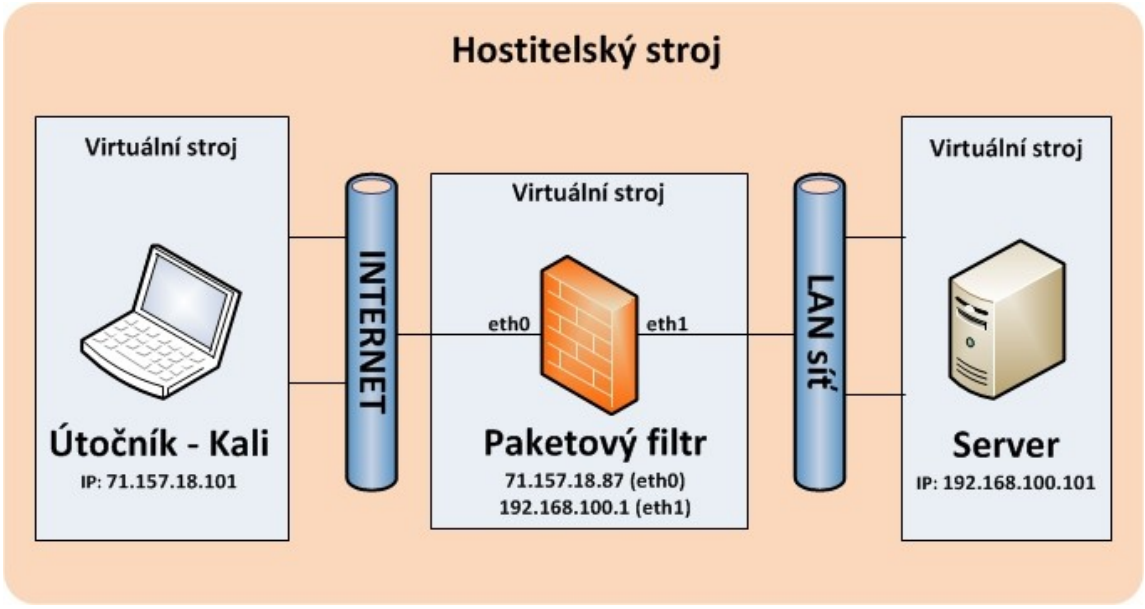
- Internet Control Message Protocol (ICMP) zprávy
- Secure Shell (SSH) relace
- Web relace přes HTTP/HTTPS

S výjimkou přístupu ke službám popsaných výše, veškerý ostatní provoz by měl být zablokován. Spojení iniciované z interní sítě nebo přímo z paketového filtru samotného by měla být sledována pomocí iptables a to včetně paketů, které tak nečiní v souladu s nastavenou bezpečnostní politikou. Kromě toho by měl paketový filtr zavést mechanismy sloužící ke kontrole podvržených paketů směrované z externí sítě na interní IP adresy. Dále pak jsem zvolil následující podmínky z pohledu paketového filtru:

- musí být dostupný pomocí protokolu SSH z interní i externí sítě,

- může akceptovat zprávy ICMP Echo z interní sítě s tím, že nežádoucí typy ICMP paketů které nejsou klasifikovány jako Echo, musí být zahozeny z jakékoliv zdrojové IP adresy,
- měl by být nastaven tak, aby jakékoliv aktivity týkající se podivných paketů, skenování portů či jiných pokusí o realizaci spojení, které nejsou povoleny, zaznamenával do logovacího souboru a zahazoval.

Při sestavování bezpečnostní politiky je nezbytné vycházet z topologie sítě, kterou máme k dispozici. Na základě této skutečnosti jsem vytvořil topologii testovací virtuální sítě (Obrázek 2.1), kterou budu používat. Topologie je použita jak v této kapitole, která hovoří o přípravě a testování zvolené bezpečnostní politiky, ale i během testů síťových útoků, které budou popsány v následující kapitole.



Obrázek 2.1: Topologie testovací virtuální sítě

Testovací síť jsem rozdělil do dvou částí. Síť LAN (interní síť) s privátní IP adresou sítě 192.168.100.0 a maskou 255.255.255.0 (nebo dle CIR notace /24) a externí síť (INTERNET). Paketový filtr má pro komunikaci mezi sítěmi použity dvě rozhraní. Interní rozhraní, označeno jako ***eth1***, s nastavenou IP adresou 192.168.100.1 slouží serveru jako výchozí brána. Toto rozhraní umožnuje serveru směrovat všechny pakety, které nejsou určeny pro jeho LAN síť, do jiné sítě. Externí rozhraní, označeno jako ***eth0***, má IP adresu 71.157.18.87 a slouží pro přístup do sítě LAN z internetu.

Skript politiky paketového filtru

Shellový skript se dá popsat jako soubor, obsahující sérii příkazů nadefinovaných administrátorem, které jsou zpracovány a spuštěny. Účelem použití skriptů je automatizace

zdlouhavých procesů, kterou ocení téměř každý administrátor. Na začátku samotného skriptu, který jsem vytvořil a pojmenoval **iptablesDP.sh**, jsou uvedeny čtyři proměnné (**3**), IPTABLES, INT_NET (IP rozsah LAN sítě), INT_INTF (rozhraní eth1) a EXT_INTF (rozhraní eth0). Jakékoliv existující pravidlo iptables je vymazáno z kernelu (**4**) a filtrovací politika pro řetězce INPUT, OUTPUT a FORWARD je nastavena na DROP (**5**).

3

```
#!/bin/sh
IPTABLES=/sbin/iptables
INT_NET=192.168.100.0/24
INT_INTF=eth1
EXT_INTF=eth0
```

4

```
$IPTABLES -F
$IPTABLES -F -t nat
$IPTABLES -X
```

5

```
$IPTABLES -P INPUT DROP
$IPTABLES -P OUTPUT DROP
$IPTABLES -P FORWARD DROP
```

Rětězec INPUT

INPUT řetězec specifikuje, zda pakety určené pro lokální systém (tedy pokud předtím kernel rozhodne na základě směrování, že je paket určen pro lokální IP adresy) mají oprávnění na přenos do sítě LAN či nikoliv. Pokud je pravidlo řetězec INPUT nastaveno na DROP, je veškeré úsilí o navázání spojení pomocí jakéhokoliv protokolu (TCP, UDP nebo ICMP) odepřeno, tedy každý paket je zahozen. Tato skutečnost se však nevztahuje na protokol ARP (Address Resolution Protocol), který je nedílnou součástí Ethernet sítí. Jelikož ARP protokol pracuje na spojové vrstvě na místo síťové vrstvy, iptables tak není schopen filtrovat tento provoz, neboť filtruje pouze IP provoz a jiné protokoly (filtrování IP paketů na základě MAC adres spojové vrstvy). Z tohoto důvodu jsou požadavky a odpovědi ARP odesílány a přijímány bez ohledu na nastavenou politiku iptables. Je ale nutné zmínit, že přenos ARP zpráv lze filtrovat pomocí arptable, ale v rámci této práce se touto problematikou zabývat nebudu. Pokračováním v definování politiky skriptu **iptablesDP.sh**, která souvisí s řetězcem INPUT, jsem zvolil následující pravidla:

6

```
###Pravidla pro sledování a zahazování stavů spojení INVALID###
$IPTABLES -A INPUT -m conntrack --ctstate INVALID -j LOG --log-prefix "DROP
INVALID " --log-ip-options --log-tcp-options
$IPTABLES -A INPUT -m conntrack --ctstate INVALID -j DROP
```

7
```
###Povolení 3-way handshake###

$IPTABLES -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

8
```
###Povolující pravidla pro spojení směrem z interní sítě ###

$IPTABLES -A INPUT -p icmp -i $INTERNI_INTF --icmp-type 8 -m conntrack --ctstate
NEW -j ACCEPT

$IPTABLES -A INPUT -p tcp -s $INTERNI_NET --dport 80 -m conntrack --ctstate NEW
-j ACCEPT

$IPTABLES -A INPUT -p tcp -s $INTERNI_NET --dport 443 -m conntrack --ctstate NEW
-j ACCEPT

$IPTABLES -A INPUT -p tcp -s $INTERNI_NET --dport 22 -m conntrack --ctstate NEW
-j ACCEPT
```

9
```
###Výchozí INPUT LOG pravidlo###

$IPTABLES -A INPUT ! -i lo -j LOG --log-prefix "DROP " --log-ip-options --log-
tcp-options
```

10
```
###Povolení pro loopback###

$IPTABLES -A INPUT -i lo -j ACCEPT
```

Připomínám, že politika odpovídá nastavení stavového paketového filtru. Tedy rozlišuje různé stavy paketů v rámci jednotlivých relací (spojení) a jeho úkolem je propustit pouze takové, které patří do již povolené relace. Pakety, které nesplní tyto podmínky, budou logovány a zahozeny. Toho je dosaženo pomocí dvou příkazů (6), které budou nepatrně pozměněny a rovněž použity pro řetězce OUTPUT a FORWARD. Stav shody (-m) je použit každým pravidlem společně s kritériem INVALID, ESTABLISHED nebo RELATED. Stav INVALID platí pro pakety, které neodpovídají žádnému známému spojením - například paket TCP FIN, který dorazí na rozhraní a není součástí žádné TCP relace, by odpovídal tomuto stavu. Paket pro stav ESTABLISHED se vztahuje ke spojení, kde probíhá obousměrná komunikace (tedy již k vytvořenému spojení). RELATED stav popisuje pakety, které vytvářejí nové spojení, ale toto spojení se již vztahuje k některému z existujících - například vracející se zpráva ICMP Port Unreachable nebo služba FTP. Pravidla pro 3-way handshake (7), které budou rovněž obdobným způsobem použity i pro řetězce OUTPUT a FORWARD, jsou přidány, aby procházející pakety mohly úspěšně navázat TCP spojení nejen se serverem. U řetězec povolující spojení (8) jsem zvolil pravidla pro spojení služby SSH z LAN sítě a zprávy ICMP Echo Request, které jsou povoleny pro jakýkoliv zdroj. Pravidlo potvrzující SSH spojení používá stav NEW, což znamená, že paket začíná vytvářet nové spojení, společně s argumentem --syn. To odpovídá pouze na pakety TCP s příznakem FIN, RST a ACK. Na závěr jsem nastavil výchozí logovací pravidlo (9), které žádnou akci s paketem neprovede, ale pouze zaznamená informace pro bližší inspekci v případě potřeby. Pravidla pro logování obsahují příkazy **--log-ip-options**, které zapíše nastavení z IP hlavičky paketu a **--log-tcp-options**, které zapíše nastavení z TCP hlavičky paketu. Tyto funkce jsou

důležité při odhalování síťových útoků. Pravidlo povolující loopback (**10**) se využívá především k vzájemné komunikaci vnitřních součástí systému.

Řetězec OUTPUT

Řetězcem OUTPUT můžeme filtrovat pakety, které jsou vytvořeny v rámci interní sítě. Například pokud je požadavek na SSH relaci vytvořen uživatelem ze strany paketového filtru, je OUTPUT řetězec použit pro povolení či zamítnutí odchozího SYN paketu. Související příkazy, které jsem nadefinoval do skriptu **iptablesDP.sh**, a jsou následující:

11

```
###Povolující pravidla pro spojení směrem do interní sítě###

$IPTABLES -A OUTPUT -p tcp -s $INTERNI_NET --dport 22 -m conntrack --ctstate NEW
-j ACCEPT

$IPTABLES -A OUTPUT -p tcp -s $INTERNI_NET --dport 80 -m conntrack --ctstate NEW
-j ACCEPT

$IPTABLES -A OUTPUT -p tcp -s $INTERNI_NET --dport 443 -m conntrack --ctstate
NEW -j ACCEPT

$IPTABLES -A OUTPUT -p icmp --icmp-type 8 -m conntrack --ctstate NEW -j ACCEPT
```

V souladu s požadavky na nastavení bezpečnostní politiky řetězce OUTPUT (**11**) budu předpokládat, že bude spojení iniciované paketovým filtrem v rámci LAN sítě za účelem kontroly dostupnosti serveru pomocí ICMP Echo zpráv. Přístupu k jeho službám HTTP nebo HTTPS či možnosti přístupu na server prostřednictvím SSH služby.

Řetězec FORWARD

Předchozí popsané nastavení filtrovací politiky striktně stanovuje schopnost paketu napřímo komunikovat mezi paketovým filtrem a serverem. Tyto pakety jsou kupříkladu určeny buď pro požadavky na SSH spojení do LAN sítě nebo lokálně iniciované spojení na paketový filtr za účelem jeho administrace prostřednictvím webového prohlížeče za pomocí služby HTTP či HTTPS. Linux může fungovat i jako směrovač, tedy předávat pakety z jednoho rozhraní na jiné, z jedné sítě do druhé. Tedy pakety, jenž nejsou určeny pro žádné zařízení v rámci LAN sítě, nýbrž jen prochází, tak putují v rámci paketového filtru řetězcem FORWARD.

12

```
###Přeposílání paketů z internetu do LAN sítě###

$IPTABLES -A FORWARD -p tcp -i $EXTERNI_INTF -o $INTERNI_INTF --dport 22
-m conntrack --ctstate NEW -j ACCEPT

$IPTABLES -A FORWARD -p tcp -i $EXTERNI_INTF -o $INTERNI_INTF --dport 80
-m conntrack --ctstate NEW -j ACCEPT

$IPTABLES -A FORWARD -p tcp -i $EXTERNI_INTF -o $INTERNI_INTF --dport 443
-m conntrack --ctstate NEW -j ACCEPT

$IPTABLES -A FORWARD -p icmp -i $EXTERNI_INTF -o $INTERNI_INTF --icmp-type 8
-m conntrack --ctstate NEW -j ACCEPT
```

13
```
###Přeposílání všech paketů z LAN sítě do internetu###

$IPTABLES -A FORWARD -i $INTERNI_INTF -o $EXTERNI_INTF -m conntrack
--ctstate NEW -j ACCEPT
```

Podobně jako pravidla nastavená pro řetězec OUTPUT, jsou spojení pro protokoly SSH, HTTP, HTTPS a ICMP povoleny (**12**) politikou řetězce FORWARD paketového filtru. Pouze tyto služby jsou povoleny z jakékoliv zdrojové IP adresy (přicházejí do paketového filtru přes rozhraní ***eth1***), jelikož chceme umožnit spojení externím uživatelům s webovým požadavkem na server umístěný v LAN síti. Tato omezení, v možnostech povolených služeb, neplatí pro směr opačný, tedy z LAN sítě do internetu. V tomto případě je povoleno jakékoliv spojení (**13**).

Nastavení NAT

Posledním krokem v návrhu bezpečnostní politiky je nastavení překladu privátních adres 192.168.100.0/24 sítě LAN, které nejsou směrovatelné v rámci sítě internet. To platí jednak pro příchozí požadavky na spojení se serverem od externích uživatelů, ale také pro odchozí spojení iniciované ze serveru v interní síti. Pro spojení z interní sítě použiji zdrojový NAT (SNAT), neboli maškarádu a pro spojení z externí sítě cílový NAT (DNAT).

14
```
###Překlad adres pro spojení z internetu do LAN sítě pro určené služby###

$IPTABLES -t nat -A PREROUTING -p tcp --dport 80 -i $EXTERNI_INTF -j DNAT
--to 192.168.100.101

$IPTABLES -t nat -A PREROUTING -p tcp --dport 443 -i $EXTERNI_INTF -j DNAT
--to 192.168.100.101

$IPTABLES -t nat -A PREROUTING -p tcp --dport 22 -i $EXTERNI_INTF -j DNAT
--to 192.168.100.101

$IPTABLES -t nat -A PREROUTING -p icmp --icmp-type 8 -i $EXTERNI_INTF -j DNAT
--to 192.168.100.101
```

15
```
###IP maškaráda (SNAT)###

$IPTABLES -t nat -A POSTROUTING -s $INTERNI_NET -o $EXTERNI_INTF -j MASQUERADE
```

S odkazem na síťový diagram obrázku 2.1, je IP adresa webového serveru v interní síti 192.168.100.101. Příkazy potřebné pro provedení překladu síťových adres jsou uvedeny výše ve výpisu použitého skriptu. Čtyři pravidla řetězce PREROUTING (**14**) umožnuje požadavkům na služby HTTP, HTTPS, ICMP a SSH být úspěšně spojeny se serverem. Pravidlo POSTROUTING (**15**) dovoluje serveru vytvořit spojení se síti internet a vystupovat tak pod IP adresou 71.157.18.87. Na závěr je nezbytné v rámci Linux kernelu povolit IP směrování (**16**) :

16
```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Uložení a spuštění bezpečnostní politiky

Díky výchozímu nastavení kernelu dojde k vymazání iptables pravidel po restartu systému. Každý administrátor, který vytváří pravidla potřebná pro bezpečnostní politiku, ocení možnost automatického načtení pravidel v případě restartu systému. Optimálním řešením se nabízí použití balíčku *iptables-persistent*, díky kterému dojde k načtení nadefinovaných pravidel a jejich aplikování při opětovném startu systému. Této skutečnosti však předchází podmínka vykonání příkazu *iptables-save* a uložení aktuálních pravidel do souboru s umístěním v /etc/iptables/rules.v4. Finálním spuštěním skriptu (**17**) dojde k vypsání stavových informací na konzoli terminálu, jenž realizuje a aktivuje stavový paketový filtr.

17

```
root@PaketovyFiltr:/# ./home/michal/Plocha/DP/iptablesDP.sh
[+] Vymazavam aktualni pravidla iptables...
[+] Nastavuji vychozi politiku retezcu...
[+] Nastavuji retezec INPUT...
[+] Nastavuji retezec OUTPUT...
[+] Nastavuji retezec FORWARD...
[+] Nastavuji pravidla NAT...
[+] Povoluji IP presmerovani...
Spusteni skriptu probehlo uspesne!!!
```

V tuto chvíli máme nastavenou funkční bezpečnostní politiku, která slouží k základní úrovni kontroly nad procházejícími pakety, stavového paketového filtru.

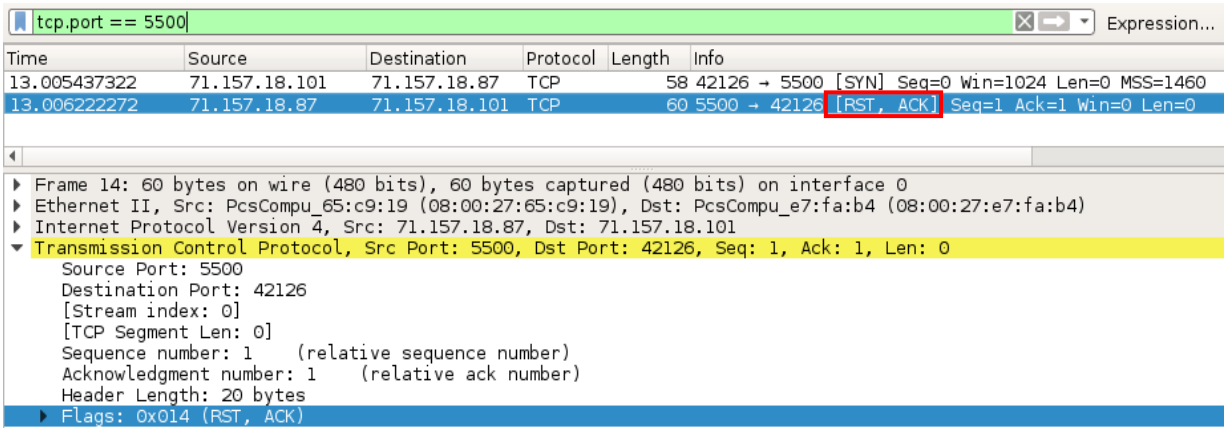
2.3 Testování zvolené bezpečnostní politiky

V tuto chvíli mám vytvořenou základní bezpečnostní politiku a ověřenou konektivitu mezi útočníkem a serverem skrze paketový filtr. Na základě těchto skutečností otestuji politiku, abych se ujistil, že odpovídá stanoveným požadavkům. Nejdůležitější ověřovací částí je test iniciovaný útočníkem v síti internet na server v lokální síti, jelikož tato situace je v praxi nejčastější a bývá tak zdrojem většiny průniků. Pro celkovou efektivitu testování v praxi by bylo rovněž vhodné provést zkoušky vedené z interní sítě, ale tento scénář předpokládat ani realizovat nebudu, jelikož není v rámci zvolené síťové topologie potřebný.

Při testování pomocí protokolů TCP a UDP využiji takzvaného skenování portů. To je proces, kterým se zjišťuje, zda je určitý TCP/UDP port otevřený nebo uzavřený. Otevřeným portem se rozumí takový port, pro který existuje program, jenž odpovídá na dotazy. Když pro daný port není spuštěn žádný program, pak je port uzavřený. Je tu ještě možnost, že pro port existuje program, který odpovídá na dotazy, ale rozhoduje se, komu odpoví a komu ne. Když se tedy pokoušíme zjistit, zda je náš port otevřený nebo uzavřený (tedy obsluhovaný programem) může se stát, že z jednoho počítače se dozvíme, že otevřený je, z jiného, že není. Takový port se označuje jako filtrovaný [12] [14].

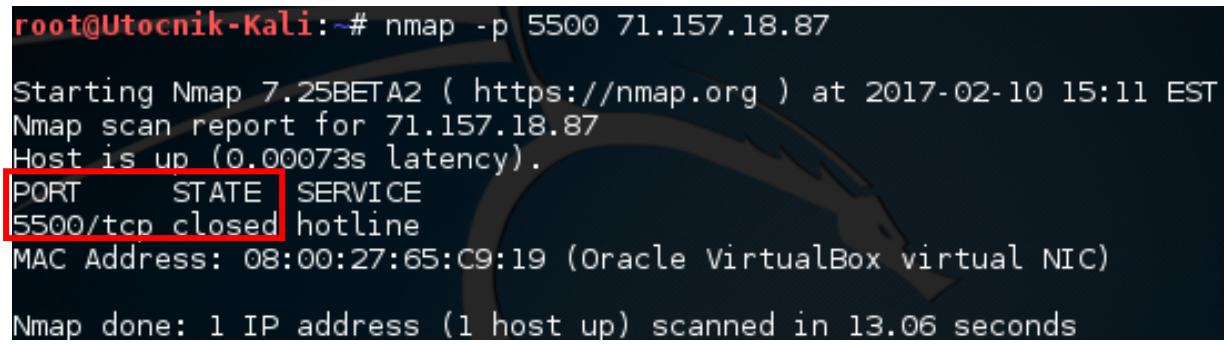
Testování politiky: TCP

Jako první v pořadí otestuji politiku skenováním náhodně zvoleného TCP portu 5500, který je iniciovaný ***útočníkem-Kali*** a neměl být na serveru dostupný. Server vygeneruje resetující paket RST/ACK, pokud je SYN paket přijat na zavřený port, a odešle jej útočníkovi. Popsaný přenos (obrázek 2.2) jsem zachytil pomocí Wireshark (modře označený řádek výstupu).



Obrázek 2.2: Zachycený přenos během skenu portu TCP/5500

Díky obrázku 2.3 lze rozpoznat, že zavřený port je přístupný (přijímá a reaguje na pakety Nmap), ale žádná aplikace momentálně na daném portu nenaslouchá. Jeho status je označen jako ***closed***. Jelikož je uzavřený port i přesto dosažitelný, může být využit útočníkem ve svůj prospěch.



Obrázek 2.3: Sken portu TCP 5500 bez použití iptables

Po aplikaci skriptu **iptablesDP.sh** snadno ověřím, že iptables blokuje tyto nežádané pakety. Dle požadavku útočník nedosáhl vytvoření TCP relace, protože byl jeho pokus s paketem TCP SYN zachycen a zahozen paketovým filtrem. Nyní však se statusem ***filtered*** (Obrázek 2.4). Nmap tak nemůže určit, zda port je otevřen, protože paketový filtr brání dosažení portu. Vhodným nastavením logovacích pravidel bezpečnostní politiky nejsou pakety pouze zahozeny, ale dojde i k vytvoření logovacích zpráv.

```
root@Utocnik-Kali:~# nmap -p 5500 71.157.18.87

Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-02-10 15:30 EST
Nmap scan report for 71.157.18.87
Host is up (0.00040s latency).
PORT     STATE    SERVICE
5500/tcp filtered hotline
MAC Address: 08:00:27:65:C9:19 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 13.26 seconds
```

Obrázek 2.4: Sken portu TCP 5500 s použitím iptables

Část filtrovaného výstupu z ***var/log/messages*** (obrázek 2.5) ukazuje, že řetězec FORWARD pracuje dle požadavku. Absence paketu RST/ACK v odpovědi na pokus o spojení naznačuje, že politika zachytila SYN paket a neumožnila tak vytvoření RST/ACK směrem zpátky k útočníkovi.

```
root@PaketovyFiltr:/home/michal/Plocha/DP# tail /var/log/messages
| grep 5500 | tail -n 1
Feb 10 21:31:09 PaketovyFiltr kernel: [17433.035360] DROP IN=eth0
OUT= MAC=08:00:27:65:c9:19:08:00:27:e7:fa:b4:08:00 SRC=71.157.18.1
01 DST=71.157.18.87 LEN=44 TOS=0x00 PREC=0x00 TTL=46 ID=3203 PROTO
=TCP SPT=47890 DPT=5500 WINDOW=1024 RES=0x00 SYN URGP=0 OPT (02040
5B4)
```

Obrázek 2.5: Výstup z logovacího souboru paketového filtru po skenu portu TCP 5500

Testování politiky: UDP

Jako druhý test uvedu schopnost politiky filtrovat pokus o skenování UDP portu. Abych ověřil, že je politika nastavena správně, použiju podobný test jako při skenování TCP portu. Nicméně tentokrát budou reakce na průběh spojení jiné. Pokud není UDP paket filtrován, měl by útočník obdržet zprávu ICMP Port Unreachable. Pomocí nástroje wireshark jsem zachytil přenos (obrázek 2.6) a provedl analýzu ICMP paketu.

(udp.port==5500) || (icmp) Expression...

Time	Source	Destination	Protocol	Length	Info
0.001976978	71.157.18.87	71.157.18.101	ICMP	113	Destination unreachable (Network unreachable)
2.503102562	71.157.18.87	71.157.18.101	ICMP	113	Destination unreachable (Network unreachable)
6.504798750	71.157.18.87	71.157.18.101	ICMP	113	Destination unreachable (Network unreachable)
9.005034791	71.157.18.87	71.157.18.101	ICMP	113	Destination unreachable (Network unreachable)
13.005759296	71.157.18.101	71.157.18.87	UDP	42	48604 → 5500 Len=0
13.106818697	71.157.18.101	71.157.18.87	UDP	42	48605 → 5500 Len=0

Obrázek 2.6: Zachycení právy ICMP Port Unreachable

Díky použití skriptu ***iptablesDP.sh*** paketový filtr opět zahodí tento pokus o skenování UDP portu 5500 a proběhlé události zaznamená do logovacího souboru, tak jak je zobrazeno na obrázku 2.7.

```
root@PaketovyFiltr:/home/michal/Plocha/DP# tail /var/log/messages
| grep 5500 | tail -n 1
Feb 10 21:54:11 PaketovyFiltr kernel: [18813.889818] DROP IN=eth0
OUT= MAC=08:00:27:65:c9:19:08:00:27:e7:fa:b4:08:00 SRC=71.157.18.1
01 DST=71.157.18.87 LEN=28 TOS=0x00 PREC=0x00 TTL=42 ID=39112 PROT
O=UDP SPT=37653 DPT=5500 LEN=8
```

Obrázek 2.7: Výstup z logovacího souboru paketového filtru po skenu portu UDP 5500

Testování politiky: ICMP

Nakonec otestuji bezpečnostní politiku ICMP skenem. Pravidla iptables, které jsem použil při sestavování politiky, využívají možnosti filtrace konkrétních typů ICMP zpráv. Omezují tak přenos jen paketů ***Echo Request***. Všechny ostatní zprávy iniciované ze sítě internet budou zahozeny. Test provedu vygenerováním ICMP zpráv Echo Reply (typ 0), jak je uvedeno na obrázku 2.8.

```
root@Utocnik-Kali:~# nping --icmp --icmp-type 0 71.157.18.87

Starting Nping 0.7.25BETA2 ( https://nmap.org/nping ) at 2017-02-10 16:02 EST
SENT (0.0030s) ICMP [71.157.18.101 > 71.157.18.87 Echo reply (type=0/code=0) id=
58289 seq=1] IP [ttl=64 id=40511 iplen=28 ]
SENT (1.0036s) ICMP [71.157.18.101 > 71.157.18.87 Echo reply (type=0/code=0) id=
58289 seq=2] IP [ttl=64 id=40511 iplen=28 ]
SENT (2.0055s) ICMP [71.157.18.101 > 71.157.18.87 Echo reply (type=0/code=0) id=
58289 seq=3] IP [ttl=64 id=40511 iplen=28 ]
SENT (3.0079s) ICMP [71.157.18.101 > 71.157.18.87 Echo reply (type=0/code=0) id=
58289 seq=4] IP [ttl=64 id=40511 iplen=28 ]
SENT (4.0096s) ICMP [71.157.18.101 > 71.157.18.87 Echo reply (type=0/code=0) id=
58289 seq=5] IP [ttl=64 id=40511 iplen=28 ]

Max rtt: N/A | Min rtt: N/A | Avg rtt: N/A
Raw packets sent: 5 (140B) | Rcvd: 0 (0B) | Lost: 5 (100.00%)
Nping done: 1 IP address pinged in 5.01 seconds
```

Obrázek 2.8: Generování zpráv ICMP Echo Reply

Nastavená politika zahodí tento pokus díky pravidlům řetězce FORWARD. Událost zaznamená do logovacího souboru a označí stavem DROP INVALID tak, jak je zobrazeno na obrázku 2.9.

```
root@PaketovyFiltr:/home/michal# tail /var/log/messages | grep ICMP | tail -n 1
Feb  4 19:38:38 Debian kernel: [ 4115.284524] DROP INVALID IN=eth0 OUT=eth1 MAC=
08:00:27:65:c9:19:08:00:27:e7:fa:b4:08:00 SRC=71.157.18.101 DST=71.157.18.87 LE
N=28 TOS=0x00 PREC=0x00 TTL=63 ID=60266 PROTO=ICMP TYPE=0 CODE=0 ID=40602 SEQ=5
```

Obrázek 2.9: Výstup z logovacího souboru paketového filtru pro ICMP sken

2.4 Monitorovací nástroj Cacti

Vhodné nastavení monitorování zařízení a jejich služeb umožnuje administrátorovi odhalit nejen aktuálně probíhající útoky na základě abnormalit, které mohou být v síti neobvyklé. Rozhodl

jsem se vytvořit si vlastní monitorovací systém pomocí nástroje Cacti [12]. Tento nástroj mne zaujal především tím, že je dostupný zdarma a umožnuje výstup v podobě přehledných grafů, volnost při vytváření monitorování formou vlastních šablon a skriptů. Díky vytvoření vlastních šablon budu mít tak detailní přehled o tom, jak monitoring probíhá, a zvolené řešení bude přesně odpovídat mým potřebám. Primárně, je sběr dat pro Cacti založený na SNMP (Simple Network Management Protocol) protokolu, který je velice užitečný pro správu počítačové sítě. Bylo tedy potřeba nainstalovat patřičné knihovny a provést nastavení SNMP služby jednak pro monitorování paketového filtru, ale i webového serveru, který se stává cílem simulovaných útoků.

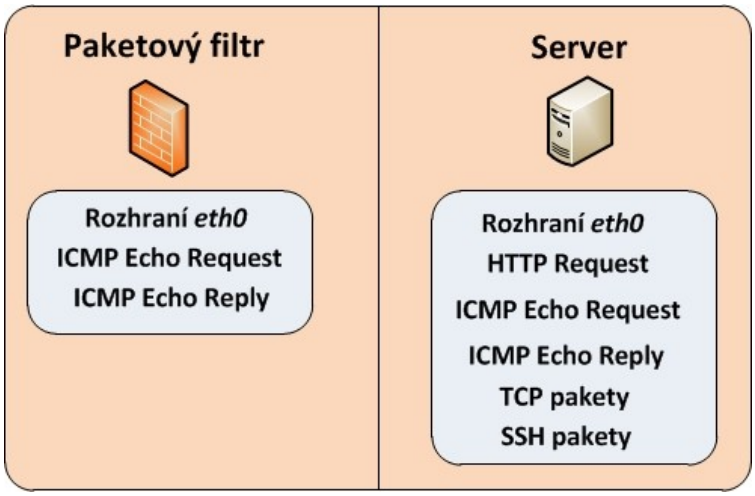
18
```
###Povolení protokolu SNMP pro Cacti monitoring ###

$IPTABLES -A OUTPUT -p udp -s $INTERNI_NET --dport 161 -m conntrack --ctstate
NEW -j ACCEPT

$IPTABLES -A INPUT -p udp -s $INTERNI_NET --dport 162 -m conntrack --ctstate
NEW -j ACCEPT
```

V neposlední řadě bylo nezbytné přidat do bezpečnostní polity skriptu **iptablesDP.sh** pravidla (**18**) určená k povolení komunikace mezi zvoleným SNMP managerem (paketový filtr) a jeho SNMP agentem, tedy serverem na portu UDP/161 a UDP/162.

Nastavením vlastní šablony docílím vytvoření grafu, zobrazující hodnoty jak v reálném čase, tak i z historického pohledu proběhlé události. Avšak informace týkající se hodnot CPU a RAM nemohou být při posuzování průběhu útoků v mém případě využívány jako směrodatné údaje, jelikož souvisejí s hodnotami hostitelského stroje jako celku. Tedy všechny vytvořené virtuální stroje využívají sdílených prostředků CPU a RAM hostitelského stroje. Nelze je proto brát jako jednotlivé určující údaje. Tyto dva indikátory budou proto vzhledem k použití virtualizace v rámci této diplomové práce zcela opomíjeny. S ohledem na zvolené síťové útoky jsem v Cacti vytvořil šablony, které budou monitorovat specifické údaje charakteristické pro simulované útoky. S pomocí těchto vytvořených šablon budu mít k dispozici aktuální přehled nad děním v síti. Zaměřil jsem se na monitorování údajů uvedených na obrázku 2.10.



Obrázek 2.10: Tabulka monitorovaných údajů

3 Průniky a vyhodnocení

DoS/DDoS útoky mají dlouhou historii a stále pro nás, jako to uživatele, představují čím dále tím větší hrozbu. Útoky samotné se staly od svého vzniku postupně sofistikovanějšími, silnějšími, cílenějšími a jejich následky mohou ovlivnit chod běžného uživatele. Proto je dle mého názoru důležité znát základní síťové charakteristiky, které nám umožní případný útok detekovat a následným způsobem jej i potlačit. V této práci se budu jen a výhradně soustředit na útoky typu DoS, tedy útoky iniciované jedním zdrojem. V tomto případě již dříve zmíněným útočníkem označeným jako Útočník-Kali.

Cílem této kapitoly je přiblížit problematiku útoků DoS/DDoS. Ověřit vliv tří zvolených typů útoků na funkcionalitu webového serveru. Jejich související popis, vyhodnocení průběhu na nezabezpečenou a zabezpečenou síť s následným zamezení útoků za použití cílených pravidel iptables. Ze záplavových útoků bude simulován ICMP flood útok prostřednictvím nástroje Hping3. Z pomalých útoků, na omezení HTTP služby webového serveru, jsem zvolil nástroj Slowloris. Pro útok hrubou silou, za účelem prolomení hesla k SSH přístupu na server, nástroj Ncrack. Všechny zmíněné útoky budou realizovány pomocí operačního systému Kali linux.

3.1 Charakteristika útoků

Útok typu Denial of Service (zkráceně DoS), má jeden zdroj za cíl odepření služby, tedy určité zamezení přístupu k systémovým zdrojům nebo zdržení operací a funkcí systému. Další forma, která může být potencionálním útočníkem využita, je nasazení distribuované formy DoS útoku, označovanou jako Distributed Denial of Service (zkráceně DDoS). Útočník využívá různý počet zdrojů tak, aby byl výsledný útok úspěšnější a pro oběť obtížněji zastavitelný. Pokud je útočník při svém DoS/DDoS útoku úspěšný, cíl jeho útoku kterým může být služba nebo síť se stane nedostupnou.

S rozvojem útoku se začala měnit i motivace útočníku. Ať už vydírání, kdy útočník požaduje platbu za ukončení útoku, špionáž, kdy útočník svým útokem odvádí pozornost, odplata a sociální protest nebo ideologicky a politicky orientovaná kybernetická válka. Dnešní útoky jsou sofistikované a jednoduché na provedení. Útoky záplavou paketů bývají nahrazovány útoky na aplikační úrovni, které nezahlcují síť, ale přímo službu. K tomu stačí zahltit oběť nižším počtem paketů a pro oběť je obtížnější útok automaticky detekovat a odvrátit. Postupně se zjednodušily i podmínky pro provedení útoku. Navzdory snahám o snížení počtu incidentů, útoky rostou rychle na své frekvenci a intenzitě [12].

3.2 Dělení DoS útoků

DoS útoky, dále jen útoky, lze dělit například podle intenzity nebo vrstvy OSI modelu. Často se také používá dělení útoků na lokální a vzdálené, či chtěné a nechtěné. Mezi základní rozdělení však patří útoky za cílem vyčerpání zdroje a útoky dle rychlosti.

Útoky s cílem vyčerpání zdrojů

Tyto útoky jde rozdělit do tří základních kategorií. A to útoky vyčerpávající síťové, serverové nebo aplikační zdroje. Útoky cílené na **vyčerpání síťových zdrojů**, se snaží zabrat co největší šířku pásma (bandwidth) pomocí velkého objemu přenosu dat. Takový přenos v malém objemu se může jevit jako legitimní a neškodný, ale při velkých objemech může mít škodlivé následky. Dokud legitimní uživatel bude chtít přistupovat do atakované sítě, bude jeho spojení buď neúspěšné, nebo příliš zpomalené. Útoky typu **vyčerpání serverových zdrojů** se snaží o vyčerpání zdrojů serveru, jako je CPU, RAM, vyrovnávací paměť atd. To znemožňuje serveru schopnost zpracovávat příchozí legitimní žádosti, zatím co svoje prostředky věnuje nelegitimním. Tyto útoky však nemusí být cílené jen na servery, ale i na zařízení typu firewall, IPS apod. Mezi poslední typ útoku jednoznačně patří **vyčerpání aplikačních zdrojů**. Jedná se o poměrně nový a rozšířený typ útoku, jehož cílem jsou aplikační protokoly. Cílený nebývá jen známy HTTP protokol, ale i HTTPS, DNS, SMTP, FTP, VoIP a jiné protokoly, které vykazují zneužitelné slabiny [12].

Útoky dle rychlosti

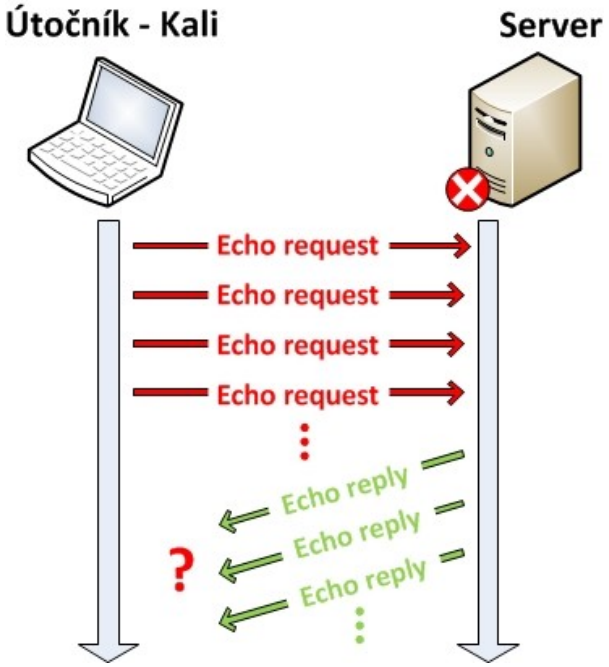
Útoky dle rychlosti se dělí na rychlé (*tzv. záplavové*) a pomalé. Jejich cílem je zahltit zdroje oběti natolik, aby nebyly obslouženy legitimní požadavky. Útočník během **záplavového** útoku zaplavuje síť masivním množstvím specifického přenosu, čímž dosahuje požadované odepření služby. Typickou charakteristikou je zvýšený počet paketů za sekundu (pps), popřípadě šířka pásma (Mbps). Velmi často se využívá ICMP, TCP a UDP protokolu. Příkladem je zde použití SYN záplavového útoku. **Pomalé** útoky, naopak, k dosáhnutí požadovaného výsledku nepotřebují objemný datový tok. Jejich specializací jsou především aplikační zranitelnosti atakovaných zařízení. Zároveň je navázané TCP spojení mezi útočníkem a jeho cílem, tedy průběh útoku se jeví jako legitimní spojení. Takovýmto chováním se vyznačuje například útok typu Slowloris, využívající časově opožděných částí nekonečné HTTP hlavičky [12]. Výše uvedené fakta napomáhají k jeho nelehké detekci a ochraně vůči jemu samému.

3.3 Popis vybraných DoS útoků

Existuje mnoho kritérií, pomocí kterých lze útoky rozpoznat. Je to dáno zejména tím, že útoky obsahují více charakteristických vlastností. Vybrané typy útoku, které budu simulovat a vyhodnocovat v této praktické části práce, nyní blíže popíši z pohledu jejich samotné funkčnosti.

ICMP flood

Útok disponující ICMP protokolem používá pakety typu ICMP Echo (Obrázek 3.1). Tyto pakety jsou využívány k zjištění dostupnosti cílového serveru. Maximální velikost tohoto paketu je sice stanovena dle doporučení RFC 791, nicméně celková velikost je škálovatelná dle potřeb útočníka. Princip je takový, že útočník odešle ICMP Echo request na cílový počítač a server mu zpátky odešle ICMP Echo reply. Je přitom zachována původní velikost paketu. Tímto způsobem je tedy linka serveru zahlcena hned dvojnásobně jednak příchozím paketem a poté odesílaným paketem [12].



Obrázek 3.1: Průběh ICMP flood útoku

S výhodou lze opět využít podvržení zdrojové IP adresy na straně útočníka, a tím se vyhnout zahlcení linky v případě zasílaných odpovědí od serveru. Útok je charakteristicky velmi jednoduchou realizací.

SSH Brute force

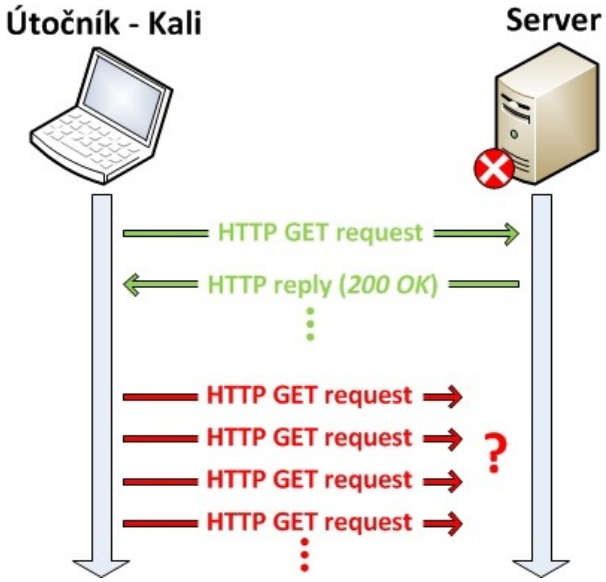
Nejčastější útoky na SSH službu jsou založeny na bázi slovníkových útoků hrubou silou. Jedná se o automatické skripty a nástroje, které zneužívají slabosti zabezpečovacího přihlašovacího mechanizmu za účelem neoprávněného vniknutí do systému. Počátek útoku spočívá v prohledávání určitého či náhodného rozsahu IP adres, kde se hledá dostupný SSH server (většinou pouze na standardním portu). Samotný útok se vyznačuje odesláním velkého počtu požadavků na přihlášení metodou pokus-omyl, při které nástroj zkouší odhadnout nejtypičtější kombinace uživatelských jmen a hesel ze souboru slovníku. Snaží se tak získat přístup na účet, dokud neprojde všechny kombinace, které má nástroj k dispozici.

Slovník může být sestaven tak, aby pokryl nejpravděpodobnější slova používané majitelem účtu vycházejícího například ze seznamu unikátních slov, který je volně k dispozici ke stažení na internetu. Tento typ útoků je na provedení velice triviální a je momentálně typem, se kterým se zcela jistě administrátor setká, pokud neomezí přístup na svůj SSH port.

Sofistikovanější útočníci pak nezkoušejí jen kombinace jmen a hesel, ale hledají starší, neaktualizované SSH servery, u kterých mohou využít dostupných objevených zranitelností [12].

Slowloris

Útok je zaměřený na znepřístupnění webových serverů a využívá při tom nízkého datového toku. Princip je takový, že si útočník udržuje mnoho spojení s webovým serverem, avšak posílá mu jen neúplné HTTP žádosti (GET). Webový server pak po určitý čas udržuje spojení a čeká na dokončení HTTP žádosti (Obrázek 3.2). Těsně před uplynutím čekací lhůty pošle útočník paket, který zajistí vynulování počítadla a čekání pokračuje. Server zůstane postupně zahlcen čekajícími spojeními.



Obrázek 3.2: Průběh Slowloris útoku

Protože počet udržovaných spojení ze strany webového serveru je přirozeně konečný, s jejich narůstajícím množstvím tak klesá rychlost napadeného webového serveru, což může vyústit v úplné odmítnutí služeb. Tento útok probíhá na aplikační vrstvě a je náročnější ho odhalit [12]. Jsou však webové servery, které jsou tímto útokem ohroženy nejvíce a to jsou Apache servery.

3.4 Simulace jednotlivých DoS útoků

Před tím, než začnu popisovat samotné průběhy a zaznamenané výsledky simulací jednotlivých DoS útoku popíši postup, kterým jsem se ubíral. Každou jednotlivou simulaci jsem rozdělil do sedmi postupných kroků:

- Volba typu DoS útoku,
- spuštění simulace útoku pomocí specifického nástroje (příkazu) na zařízení Útočník-Kali,
- útok na server přes nezabezpečenou síť,
- zaznamenání a analýza grafů,
- aplikace iptables skriptu k zamezení průniku do sítě,
- útok na server přes zabezpečenou síť,
- zaznamenání a analýza grafů.

3.4.1 ICMP Flood

Jako první v pořadí jsem provedl simulaci DoS útoku typu ICMP flood. Z obrázku 3.3 je patrné, že byl zvolen typ ICMP zprávy číslo 8, tedy *Echo Request*. Směrovaný na IP adresu 71.157.18.87, jenž odpovídá NAT adrese paketového filtru potřebnému k dosažení webového serveru v privátní síti. Přívlastkem v použité syntaxi příkazu hping3 je volba záplavového útoku, tedy *--flood.*

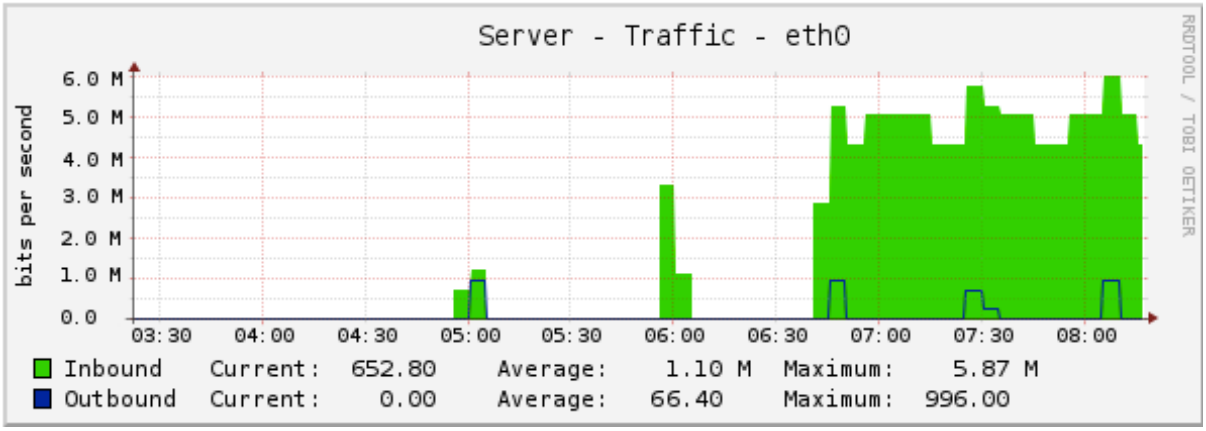
```
root@Utocnik-Kali:~# hping3 --icmp -C 8  --flood 71.157.18.87
HPING 71.157.18.87 (eth0 71.157.18.87): icmp mode set,72830 headers+,0 data bytes
hping in flood mode, no replies will be shown
^C
--- 71.157.18.87 hping statistic ---
140193872815711871002 98 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

Obrázek 3.3: Simulace ICMP flood útoku pomocí hping3

Po ukončení trvání simulace útoku byla zobrazena hodnota 14019387281571187100298, která odpovídá počtu přenesených paketů. Tato hodnota lze považovat za orientační údaj, jelikož se po uplynutí určité doby již nadále nezvyšovala. I tak však poukazuje na abnormálně zvýšené množství odeslaných paketů útočníkem. Výsledná naměřená ztrátovost paketů se pohybuje na úrovni 100 %.

Nezabezpečená síť

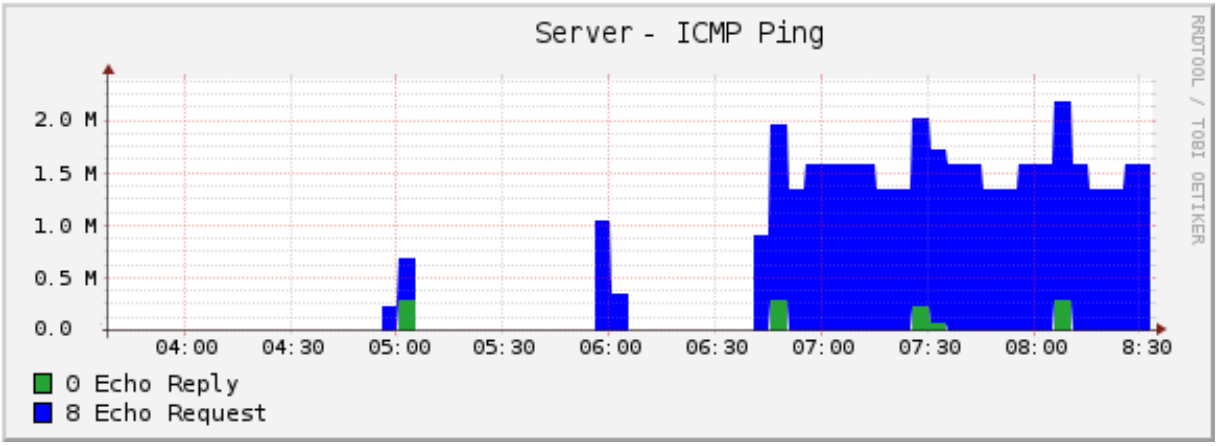
Během útoku jehož doba trvání odpovídala přibližně devadesáti minutám, prostřednictvím nezabezpečené sítě, bylo na serveru zaznamenáno skokové zvýšení hodnoty přenosové rychlosti pro rozhraní eth0 (obrázek 3.4). Naměřená příchozí hodnota se pohybovala kolem 6 Mbit/s. Odchozí hodnota se pohybovala v jednotkách menších než 1 Mbit/s.



Obrázek 3.4: Datový tok rozhraní eth0 serveru během ICMP flood útoku na nezabezpečenou síť.

Z pohledu grafu zobrazujícího počet příchozích a odchozích ICMP zpráv na server (obrázek 3.5) je znatelné, že se server potýká s abnormálním počtem příchozích ICMP Echo Request zpráv. Tyto zprávy zahltí cílový server do té míry, že není schopen adekvátně reagovat na příchozí

požadavky. Server tak reaguje nepravidelným odesláním zprávy ICMP Echo Reply v množství řádu desítek se zaznamenaným nepravidelným časovým intervalem téměř co půl hodiny.



Obrázek 3.5: Počet ICMP Echo Request/Reply zpráv na serveru během ICMP flood útoku na nezabezpečenou síť.

Jistou ochranou proti ICMP flood útoku může být kompletní zamezení průchodu ICMP Echo paketů. Nicméně tento krok považuji za poněkud radikální, neboť se může stát značně omezujícím v případě monitorování dostupnosti daného serveru z pohledu nastaveného kontrolního systému. Server se bude jevit jako nefunkční (vypnutý) i když tato informace nebude odpovídat skutečnosti.

Pravidla k zamezení ICMP flood útoku

Ze základního řetězce FORWARD odkloním příchozí ICMP pakety na nově vytvořené řetězce ICMP_flood a ICMP (**19**). Jisté zabezpečení může poskytnout modul *limit*, který umožňuje omezit počet spojení. Platí zde prvotní podmínka pro maximální počet 150 ICMP paketů (*--limit-burst 150*). V případě, že je počet příchozích ICMP paketů vyšší než dovoluje stanovený limit, pravidlo povolí pouze 3 příchozí pakety za sekundu (*--limit 3/second*). Ve zbylých případech je spojení logováno a následně zahozeno.

19

```
###Ochrana proti ICMP flood utoku###

$IPTABLES -N ICMP_flood

$IPTABLES -N ICMP


$IPTABLES -A FORWARD -p icmp -i $EXTERNI_INTF -o $INTERNI_INTF -m conntrack
--ctstate NEW -j ICMP_flood

$IPTABLES -A ICMP_flood -p icmp -m limit --limit  3/second --limit-burst 150
-j ICMP

$IPTABLES -A ICMP_flood -p icmp -m limit --limit 6/hour --limit-burst 1
-j LOG --log-prefix 'PING-DROP: ICMP FLOOD ATTACK ' --log-ip-options
--log-tcp-options

$IPTABLES -A ICMP_flood -p icmp -j DROP
```

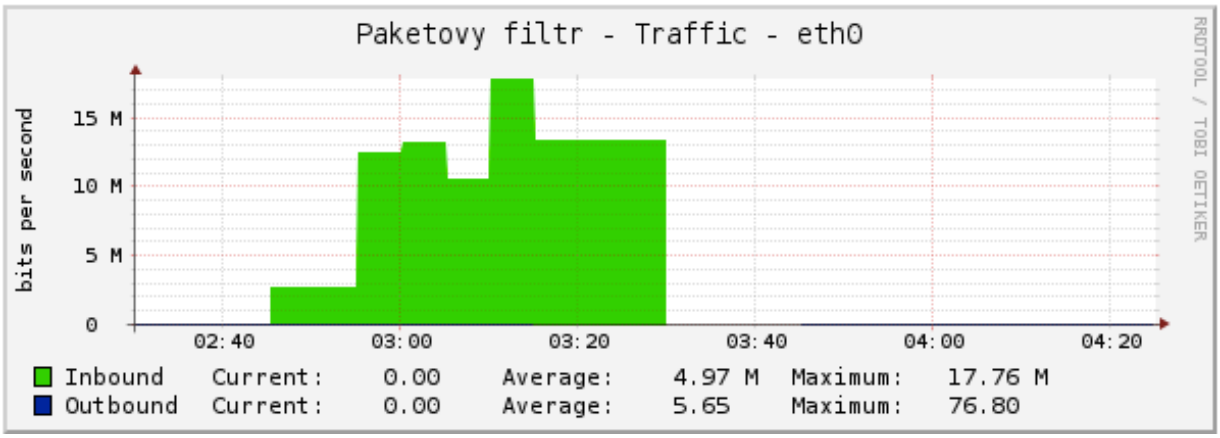
20

```
$IPTABLES -A ICMP -p icmp --icmp-type destination-unreachable -j ACCEPT
$IPTABLES -A ICMP -p icmp --icmp-type source-quench -j ACCEPT
$IPTABLES -A ICMP -p icmp --icmp-type parameter-problem -j ACCEPT
$IPTABLES -A ICMP -p icmp --icmp-type time-exceeded -j ACCEPT
$IPTABLES -A ICMP -p icmp --icmp-type echo-reply -j ACCEPT
$IPTABLES -A ICMP -p icmp --icmp-type echo-request -j ACCEPT
$IPTABLES -A ICMP -p icmp --icmp-type address-mask-request -j REJECT
--reject-with icmp-admin-prohibited
$IPTABLES -A ICMP -p icmp --icmp-type timestamp-request -j REJECT
--reject-with icmp-admin-prohibited
$IPTABLES -A ICMP -p icmp -j DROP
```

Pokud je však stanovený limit dodržen, jsou ICMP pakety přesměrovány do řetězce s názvem ICMP (**20**) a následně rozlišeny a vyhodnoceny dle nastavených pravidel pro jednotlivé typy ICMP zpráv. Lze říci, že tyto pravidla představují jistou dvoustupňovou ochranu. Tedy musejí být v případě potvrzeného požadavku na spojení splněny dvě určené podmínky.

Zabezpečená síť

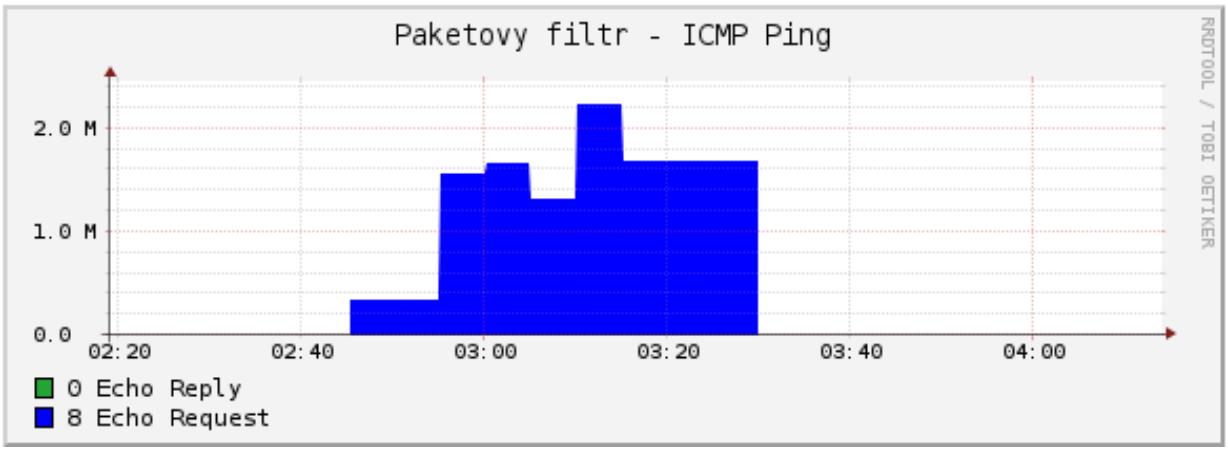
Následným opakováním simulace útoku na již zabezpečenou síť, jenž probíhal v tomto případě přibližně po dobu čtyřiceti minut, se situace dle monitorovaných údajů značně liší.



Obrázek 3.6: Datový tok rozhraní eth0 paketové filtru během ICMP flood útoku na zabezpečenou síť.

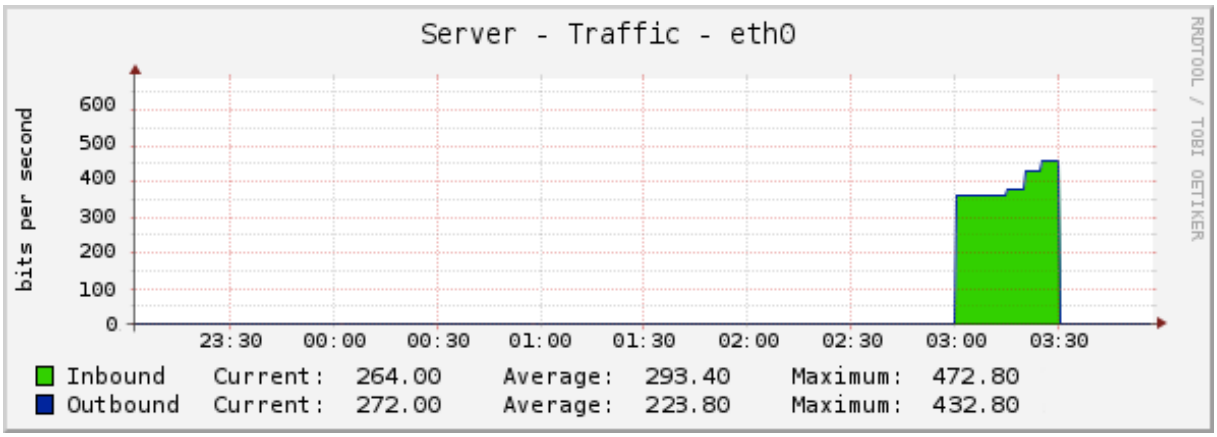
Na obrázku 3.6 je vidět orientační naměřená přenosová rychlost kolem hodnoty 15 Mbit/s pro rozhraní eth0 paketového filtru. Ten po aplikaci pravidel pro potlačení ICMP flood útoku začal filtrovat přenos ICMP zpráv dle našeho požadavku. S tím rovněž související graf na obrázku 3.7, který ukazuje množství příchozích Echo Request zpráv na paketový filtr. Avšak bez možnosti rozpoznat

odchozí Echo Reply zprávy, které se pohybují na povolené úrovni, která je však nižší než zobrazuje graf.



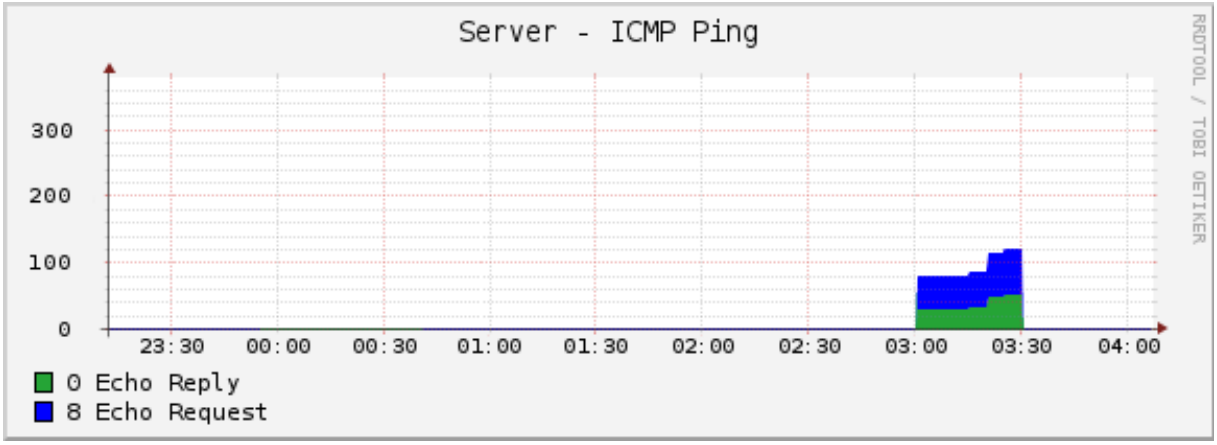
Obrázek 3.7: Počet ICMP Echo Request/Reply zpráv na paketovém filtru během ICMP útoku na zabezpečenou síť.

Funkčnost a efektivitu aplikovaných pravidel *iptables* souvisejících s filtrováním a omezením počtu ICMP Echo Request zpráv cílených na atakovaný server, lze však rozpoznat díky obrázku 3.8. Tento graf ukazuje nepatrné množství naměřené příchozí i odchozí přenosové rychlosti rozhraní eth0 serveru, pohybujících se v řádech desetin Mbit/s.



Obrázek 3.8: Datový tok rozhraní eth0 serveru během ICMP flood útoku na zabezpečenou síť.

Rovněž z obrázku 3.9 je prokazatelné, že server během útoku obdržel jen nepatrný počet ICMP Echo Request zpráv s tím, že je schopen na ně adekvátně reagovat odpovědí zprávou ICMP Echo Reply. Počet obou typů zpráv odpovídá přibližně stejnému počtu, jenž je uveden na grafu.



Obrázek 3.9: Počet ICMP Echo Request/Reply zpráv na serveru během ICMP flood útoku na zabezpečenou síť.

Z provedených simulací a souvisejících grafů z průběhu monitorování sledovaných údajů mohu konstatovat, že aplikací skriptu docílím filtraci ICMP zpráv na paketovém filtru dle požadavku. Server je z pohledu sítě dostupný i v případě působení útoku typu ICMP flood.

3.4.2 SSH Brute force

Simulace druhého útoku byla realizována prostřednictvím nástroje s názvem Ncrack. Potřebné nastavení lze vidět na obrázku 3.10. Byl zvolen cílový port pro SSH (*-p 22*), uživatelské jméno (*root*), použitý slovník (*500-worst-passwords.txt*) a destinační IP adresa. Z obrázku 3.10 je patrné, že během útoku trvajícího pouhých 84 sekund došlo ke zjištění potřebného uživatelského jména a hesla k přístupu na server. Získané údaje byly použity pro ověření přístupu na cílený server s pozitivním výsledkem.

```
root@Utocnik-Kali:~/Downloads# ncrack -p 22 --user root -P 500-worst-passwords.txt
71.157.18.87

Starting Ncrack 0.5 ( http://ncrack.org ) at 2017-03-26 08:51 EDT

Discovered credentials for ssh on 71.157.18.87 22/tcp:
71.157.18.87 22/tcp ssh: 'root' 'mik'

Ncrack done: 1 service scanned in 84.01 seconds.

Ncrack finished.
root@Utocnik-Kali:~/Downloads# ssh -l root 71.157.18.87
root@71.157.18.87's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@Server:~#
```

Obrázek 3.10: SSH Brute force útok pomocí nástroje Ncrack

Nezabezpečená síť

Jistými indiciemi, které mohou administrátorovi napomoct k rozpoznání probíhajícího typu útoku na server, jsou dohledatelné (například pomocí příkazového řádku) v samotných logovacích souborech daného serveru (obrázek 3.11).

```
root@Server:/home/michal# tail -20 /var/log/auth.log | grep 'sshd'
Mar 26 15:42:04 Server sshd[22974]: Failed password for root from 71.15
7.18.101 port 49974 ssh2
Mar 26 15:42:04 Server sshd[22974]: Connection closed by 71.157.18.101
[preauth]
Mar 26 15:42:05 Server sshd[22977]: Failed password for root from 71.15
7.18.101 port 49978 ssh2
Mar 26 15:42:05 Server sshd[22977]: Connection closed by 71.157.18.101
[preauth]
Mar 26 15:42:05 Server sshd[22981]: Failed password for root from 71.15
7.18.101 port 49986 ssh2
Mar 26 15:42:05 Server sshd[22981]: Connection closed by 71.157.18.101
[preauth]
Mar 26 15:42:05 Server sshd[22983]: Failed password for root from 71.15
7.18.101 port 49992 ssh2
Mar 26 15:42:05 Server sshd[22983]: Connection closed by 71.157.18.101
[preauth]
Mar 26 15:42:05 Server sshd[22978]: Failed password for root from 71.15
7.18.101 port 49980 ssh2
Mar 26 15:42:05 Server sshd[22978]: Connection closed by 71.157.18.101
```

Obrázek 3.11: Výstup z logovacího souboru autentifikace serveru během SSH Brute force útoku

Díky výstupu z logovacího souboru pro autentifikaci lze jednoznačně rozpoznat zvýšené množství pokusů o přístup na server prostřednictvím služby SSH. Na základě tohoto zjištění jsem vyfiltroval pomocí nástroje *netstat* v příkazovém řádku serveru počet aktuálních požadavků na dané spojení včetně jejich IP adres (obrázek 3.12). Aktuální zaznamenaný počet 227 spojení z jedné IP adresy 71.157.18.101 naznačuje probíhající DoS útok cílený na službu SSH serveru.

```
root@Server:/home/michal# netstat -ntu | grep :22 | awk '{print $5}'
|cut -d: -f1| sort | uniq -c | sort -rn
    227 71.157.18.101
```

Obrázek 3.12: Počet zjištěných aktuálních spojení během SSH Brute force útoku

Pravidla k zamezení SSH Brute Force útoku

Netfilter za pomocí modulu iptables umožnuje dynamicky vkládat uživatele na černou listinu (dále jen blacklist) za předpokladu splnění administrátorem určených pravidel. Tato možnost je velice užitečná v případě zamezení Brute force útoku na zařízení se službou SSH.

21

```
###Ochrana proti SSH Brute force utoku###

$IPTABLES -N SSH-IN

$IPTABLES -N SSH-OUT


$IPTABLES -A FORWARD -p tcp -i $EXTERNI_INTF -o $INTERNI_INTF -m conntrack
--ctstate NEW --dport 22 -j SSH-IN

$IPTABLES -A FORWARD -p tcp -i $INTERNI_INTF -o $EXTERNI_INTF -m conntrack
--ctstate ESTABLISHED,NEW,RELATED --sport 22 -j SSH-OUT


$IPTABLES -N SSH-BLACKLIST

$IPTABLES -A SSH-BLACKLIST -m recent --name blacklist --set

$IPTABLES -A SSH-BLACKLIST -m limit --limit 1/minute --limit-burst 100
-j LOG --log-prefix "SSH-BLACKLIST: " --log-ip-options --log-tcp-options

$IPTABLES -A SSH-BLACKLIST -j DROP


$IPTABLES -N SSH-WHITELIST

$IPTABLES -A SSH-WHITELIST -s 192.168.100.1 -j ACCEPT

$IPTABLES -A SSH-WHITELIST -j RETURN


###Pravidla pro prichozi SSH pozadavky###

$IPTABLES -A SSH-IN -j SSH-WHITELIST

$IPTABLES -A SSH-IN -m recent --update --name blacklist --seconds 43200
--hitcount 1 -j DROP

$IPTABLES -A SSH-IN -m recent --set --name short

$IPTABLES -A SSH-IN -m recent --set --name long

$IPTABLES -A SSH-IN -m recent --update --name short --seconds 15 --hitcount 5
-j SSH-BLACKLIST

$IPTABLES -A SSH-IN -m recent --update --name long --seconds 3600 --hitcount 30
-j SSH-BLACKLIST

$IPTABLES -A SSH-IN -j ACCEPT
```

Vytvořil jsem pomocné řetězce, které nyní popíši pro objasnění funkčnosti skriptu. Řetězce SSH_IN a SSH_OUT rozdělí veškeré SSH pakety, jenž vstoupily do základního řetězce FORWARD, do požadovaných směrů (**21**). Řetězec SSH, kam směřují nová spojení s cílovým portem TCP/22, řetězec SSH-WHITELIST, kde jsou specifikovány IP adresy a rozsahy, které mají být povoleny bez filtrování (IP adresa 192.168.100.1 je nastavena pro rozhraním *eth1* paketové filtru). Řetězec SSH-BLACKLIST provádí dvě podstatné úlohy. Jednak zařazuje danou IP adresu na ***blacklist*** a příslušný paket zahazuje, ale ještě před tím, než paket zahodí, jej uloží do logovacího souboru.

V tomto případě omezuje nastavení počet informací uložených v logovacím souboru na průměrně jednu za minutu (s možností najednou pojmout až 100 informací). Podstatné je, že do tohoto řetězce se příslušný paket dostane pouze jednou, a sice když je zařazen na blacklist. V ostatních případech jej zahazuje první pravidlo s určením (*-m recent*) v řetězci SSH. Pro blacklist jsou stanovena dva různé časové intervaly. Pro 5 spojení za 15 sekund a 30 spojení za 3600 sekund (tedy hodinu). Pokud se IP adresa pokusí o více spojení za daný časový úsek, bude zařazena na blacklistu. Doba, po kterou pak v blacklistu zůstane, je minimálně 43200 sekund (1 den). Daná IP adresa bude vyjmuta, pokud se během onoho jednoho dne od zařazení do blacklistu už nepokusí o přístup k SSH službě. Pokud to zkusí znovu během této doby, počítadlo se vynuluje a začíná se znovu.

22
```
###Pravidla pro odchozi SSH pozadavky###
$IPTABLES -A SSH-OUT -m recent --name blacklist --rdest --rcheck --seconds 30
-j SSH-REJECT
$IPTABLES -A SSH-OUT -j ACCEPT


###Řetězec pro zahazování paketů souvisejících s SSH Brute Force útokem###
$IPTABLES -N SSH-REJECT
$IPTABLES -A SSH-REJECT -p tcp -j REJECT --reject-with tcp-reset
```

Odchozí pakety související s uživatelem, který byl zaznamenán v blacklistu během posledních 30 sekund, jsou odmítnuty pomocí řetězce SSH-REJECT (**22**).

Zabezpečená síť

Opětovným spuštěním útoku, již na zabezpečenou síť, a následným zobrazením aktuální hodnoty počtu požadavků na SSH spojení se serverem si ověřím požadovanou funkčnost pravidel. Na obrázku 3.13 lze vidět zachycený počet pouhých 4 příchozích požadavků na spojení z útočníkovy IP adresy 71.157.18.101. Tato hodnota potvrzuje společně s požadavkem na jedno spojení z IP adresy 192.168.100.1 paketového filtru, který jsem během probíhajícího útoku inicioval, korektní funkčnost řetězců WHITELIST a BLACKLIST a dostupnost serveru i po dobu trvání útoku.

```
root@Server:/home/michal# netstat -ntu | grep :22 | awk '{print $5}'
|cut -d: -f1| sort | uniq -c | sort -rn
      4 71.157.18.101
      1 192.168.100.1
```

Obrázek 3.13: Počet zjištěných aktuálních požadavků na spojení během SSH Brute force útoku na zabezpečenou síť

Navrhnuté řešení k zabezpečení přístupu na server prostřednictvím služby SSH, na úrovni stavového paketového filtru, představuje díky dosaženým výsledkům funkční a přijatelnou ochranu.

3.4.3 Slowloris

Jako třetí útok v pořadí, který budu simulovat, popíši DoS útok s názvem Slowloris. Tento typ útoku si klade za cíl znepřístupnění webové služby prostřednictvím protokolu HTTP na atakovaném serveru. Pomocí skriptu slowloris.pl jsem nastavil požadované vlastnosti (obrázek 3.14).

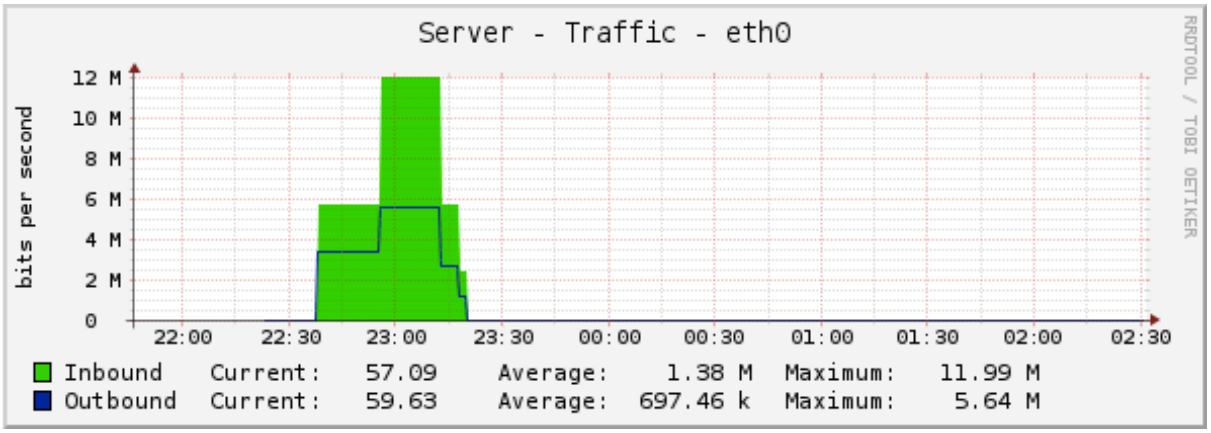
```
root@Utocnik-Kali:~/Downloads# ./slowloris.pl -dns 71.157.18.87  -port 80 -timeout 1 -num 1000
Welcome to Slowloris - the low bandwidth, yet greedy and poisonous HTTP client by Laera Loris
Defaulting to a 5 second tcp connection timeout.
Multithreading enabled.
Connecting to 71.157.18.87:80 every 1 seconds with 1000 sockets:
                Building sockets.
                Building sockets.
                Building sockets.
                Building sockets.
                Sending data.
Current stats:  Slowloris has now sent 557 packets successfully.
This thread now sleeping for 1 seconds...
```

Obrázek 3.14: Simulace Slowloris útoku pomocí skriptu slowloris.pl

Bylo nutné uvést cílovou IP adresu, v mém případě opět NAT IP adresu serveru. Společně s atributy týkajících se cílového portu TCP/80 (HTTP), s časovým limitem odesílání paketů 1 sekunda (-timeout 1) a počtem 1000 paketů (-num 1000). Hodnota odesílaných paketů je v tomto případě pouze orientační vstupní údaj potřebný ke spuštění skriptu. Při časově dlouhodobém testování se ukázalo, že útok trvá do té doby, dokud není skript manuálně zastaven specifickým příkazem.

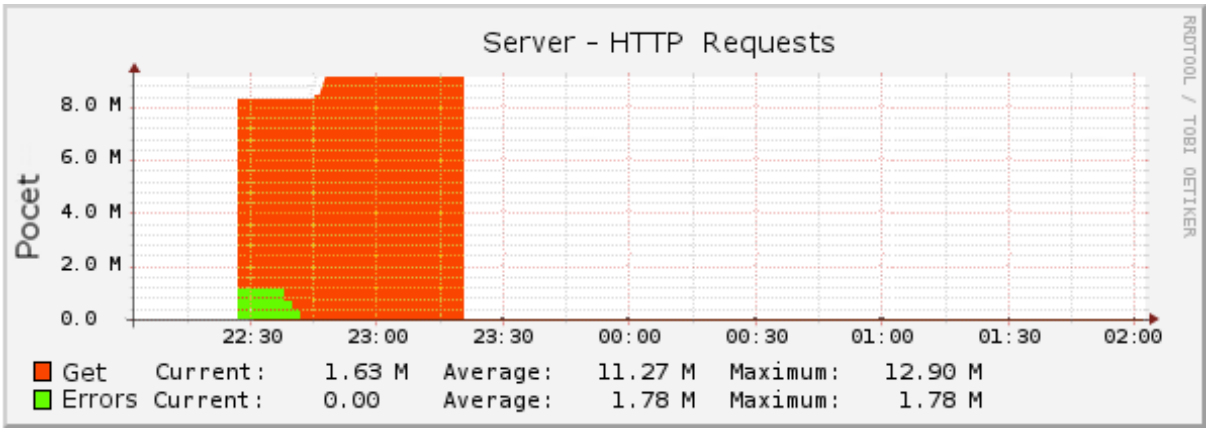
Nezabezpečená síť

Při spuštění skriptu došlo okamžitě k zaznamenání nárůstu datového toku rozhraní serveru ve směru příchozím na hodnotu téměř 12 Mbit/s (obrázek 3.15). Během testu, který trval přibližně padesát minut, tato hodnota postupně narůstala. Maximální přibližná zaznamenaná hodnota přenosové rychlosti ve směru odchozím je 5,64 Mbit/s.



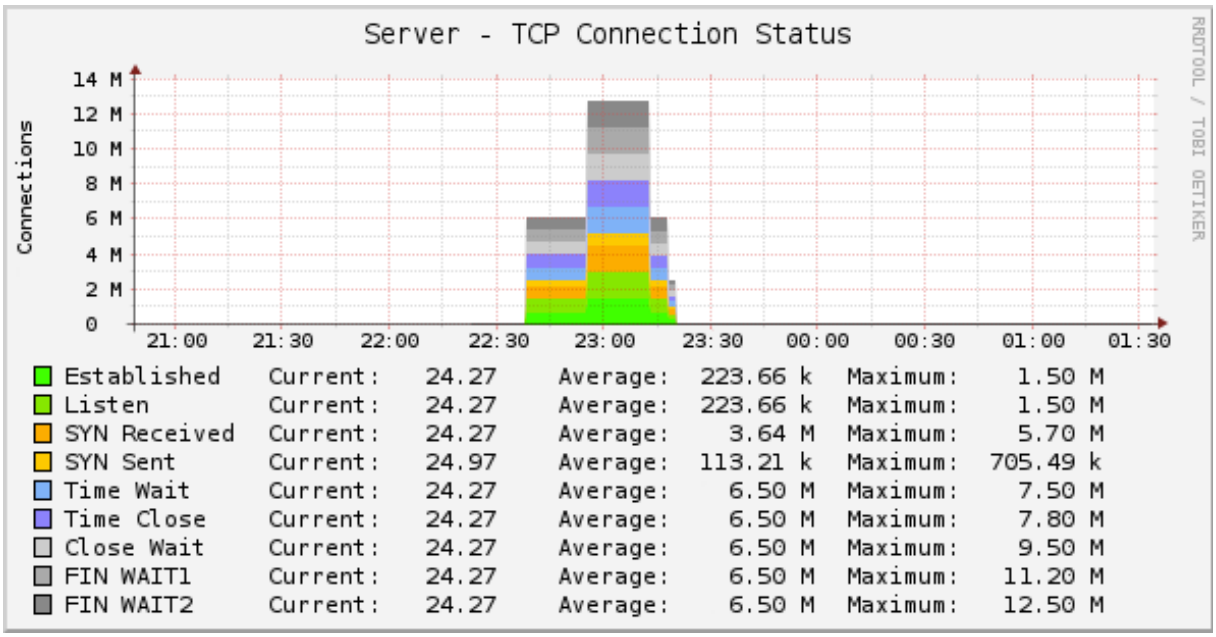
Obrázek 3.15: Datový tok rozhraní eth0 serveru během útoku Slowloris na nezabezpečenou síť.

Rovněž abnormální nárůst počtu příchozích HTTP Get zpráv na serveru (obrázek 3.16), je administrátorovi jasným ukazatelem netypického chování v síti. Myslím si, že informace o maximální zaznamenané hodnotě počtu 12,9 milionu požadavků na server a celkovém průběhu jasně poukazuje na útok cílící na službu HTTP.



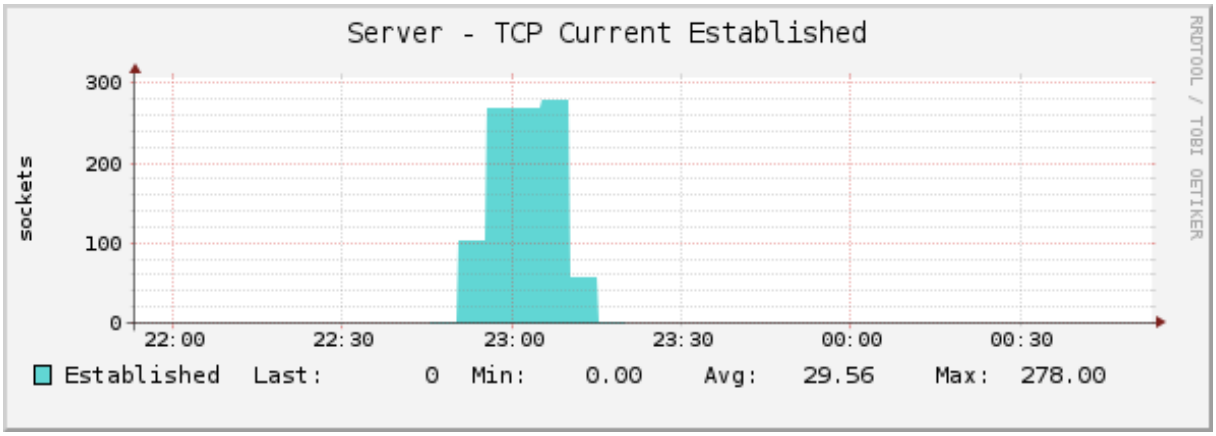
Obrázek 3.16: Počet příchozích HTTP Get/Error zpráv na server během útoku Slowloris na nezabezpečenou síť.

Vytvořením šablony v Cacti poukazující na aktuální počet a typ TCP paketů na serveru, lze docílit zobrazení jeho reakcí. Z obrázku 3.17 jde vidět, že bylo na server po dobu simulace útoku směrováno téměř 13 milionu paketů. Server byl schopen odeslat přibližně jen 705 tisíc paketů typu TCP SYN Sent.



Obrázek 3.17: Počet a typ TCP paketů na serveru během Slowloris útoku na nezabezpečenou síť.

Rovněž maximální počet současně možných navázaných TCP spojení byl překročen a zaznamenán na grafu (obrázek 3.18). Ve výchozím nastavení má Apache pro parametr sockets určenou hodnotu 256 jako to pevný limit z bezpečnostních důvodů. Chtěl bych dodat, že spuštěním nástroje Slowloris se webová stránka na serveru stala takřka okamžitě zcela nedostupnou jak z pohledu útočníka-Kali, tak samotného paketového filtru, a to po celou dobu probíhající simulace.



Obrázek 3.18: Počet sestavených TCP spojení na serveru během útoku Slowloris na nezabezpečenou síť.

Pravidla k zamezení Slowloris útoku

V případě příchozích HTTP požadavků pro více než 150 spojení za sekundu či překročení počtu větším než je 50 spojení za sekundu ze stejného IP subnetu, jsou související pakety neprodleně zahozeny. Pokud je však zaznamenán počet spojení přesahující hodnotu 5, jsou pakety s cílovým portem TCP/80 přesměrovány do nově vytvořeného řetězce s názvem Slowloris (**22**).

22

```
###Ochrana proti Slowloris (HTTP flood) utoku###

$IPTABLES -N Slowloris

$IPTABLES -N LOG-Slowloris


###Regulace poctu HTTP spojení na server###

$IPTABLES -A FORWARD -p tcp --syn -i $EXTERNI_INTF -o $INTERNI_INTF --dport 80
-m connlimit --connlimit-above 150 -j REJECT --reject-with tcp-reset

$IPTABLES -A FORWARD -p tcp --syn -i $EXTERNI_INTF -o $INTERNI_INTF --dport 80
-m connlimit --connlimit-above 50 --connlimit-mask 24 -j REJECT --reject-with
tcp-reset

$IPTABLES -A FORWARD -p tcp --syn -i $EXTERNI_INTF -o $INTERNI_INTF --dport 80
-m connlimit --connlimit-above 5 -j Slowloris

$IPTABLES -A FORWARD -p tcp --syn -i $EXTERNI_INTF -o $INTERNI_INTF
--dport 80 -j ACCEPT
```

Tento řetězec obsahuje pravidla s cílem redukovat spojení během útoku na HTTP službu serveru. Řetězec s názvem LOG_Slowloris slouží k zaznamenání logovacích zpráv, s intervalem každých deset minut (*--limit 6/hour*) určených pro administrátora k identifikaci útoku (**23**).

23

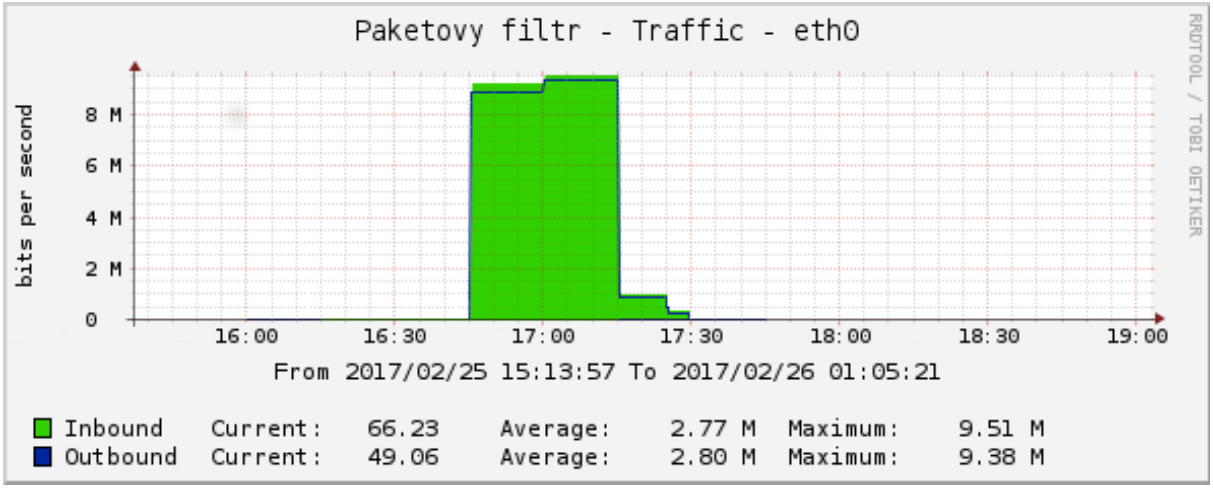
```
###Retezec SLOWLORIS urcuje pravidla pro redukci poctu HTTP spojeni na server###
$IPTABLES -A Slowloris -m limit --limit 2/second --limit-burst 80 -j ACCEPT
$IPTABLES -A Slowloris -m limit --limit 12/hour --limit-burst 1 -j LOG-Slowloris
$IPTABLES -A Slowloris -j REJECT --reject-with tcp-reset

###Retezec LOG_Slowloris zaznamenava a zahazuje nezadane pakety###
$IPTABLES -A LOG_Slowloris -j LOG --log-prefix "SLOWLORIS: HTTP-FLOOD: "
--log-tcp-options --log-ip-options -m limit --limit 6/hour
$IPTABLES -A LOG_Slowloris -p tcp -j REJECT --reject-with tcp-reset
-m comment --comment "Reject TCP connections with tcp-reset"
```

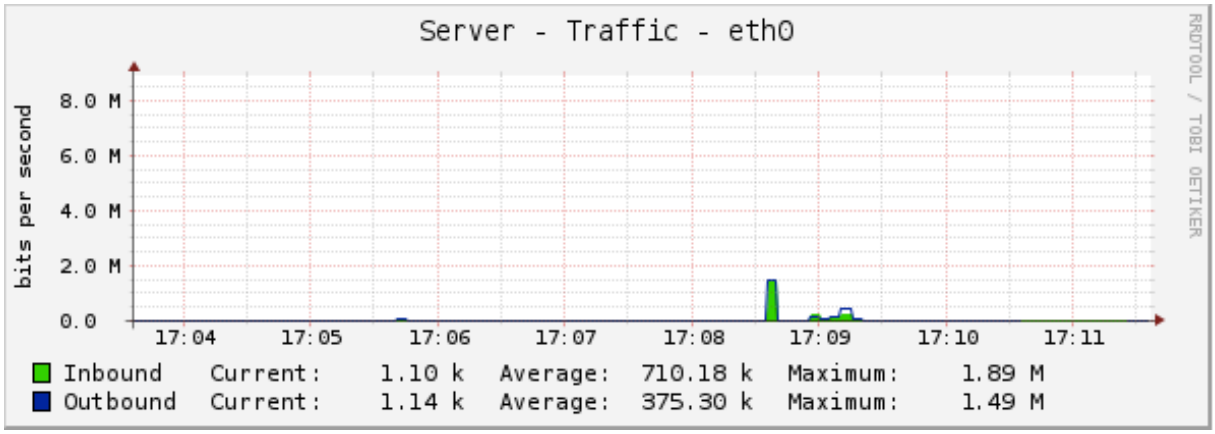
Chtěl bych opodstatnit důvod zvolení možnosti odeslání tcp-reset (*--reject-with tcp-reset*) pro pravidla související s odmítnutím paketu. Pokud zvolíme možnost odmítnutí (REJECT) spojení, webový prohlížeč zobrazí téměř okamžitě uživateli informaci, že stránka „není k dispozici“. Zato volba pravidla s možností zahození (DROP) spojení způsobí, že klient musí čekat určitou dobu a pokoušet se o spojení před tím, než dostane informaci o nedostupnosti stránky. Tato skutečnost má zavádějící charakter, protože se chová spíše jako problémy související se špatným připojením, než jako nedostupnost serveru.

Zabezpečená síť

Opětovným spuštění simulace útoku, nyní za použití cílených filtrovacích pravidel pomocí iptables, došlo k zaznamenávání požadovaných údajů po dobu zhruba třiceti minut s následujícími výsledky.

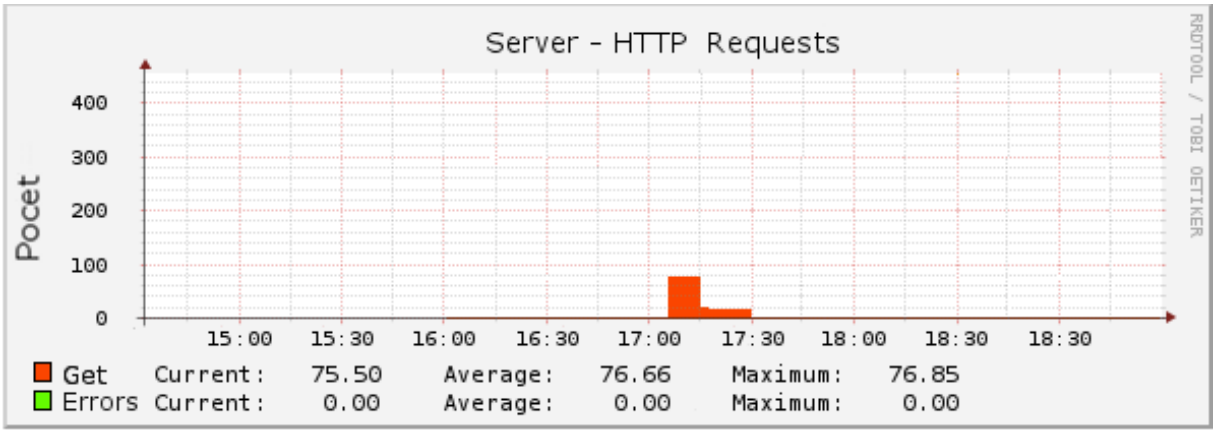


Obrázek 3.19: Datový tok rozhraní eth0 paketového filtru během Slowloris útoku na zabezpečenou síť.



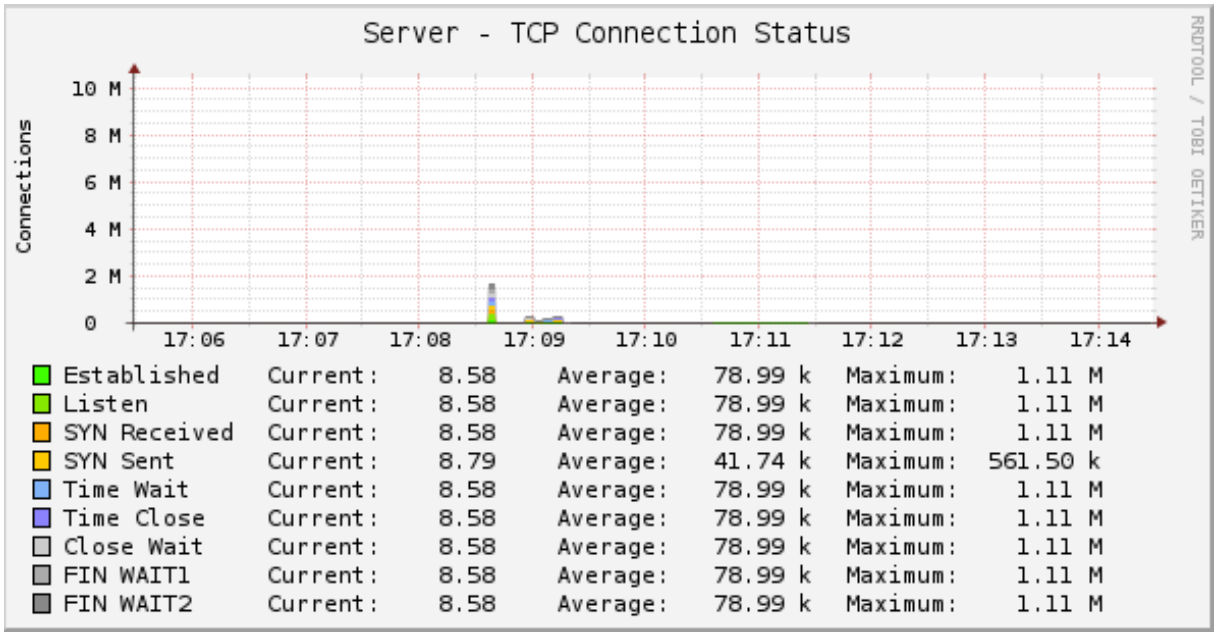
Obrázek 3.20: Datový tok rozhraní eth0 serveru během Slowloris útoku na zabezpečenou síť.

Při srovnání grafů monitorujících rozhraní eth0 paketového filtru (obrázek 3.19) a rozhraní eth0 serveru (obrázek 3.20) je rozpoznat znatelný rozdíl v zaznamenaném přibližném využití přenosové rychlosti. V tom smyslu, že paketový filtr provádí filtraci HTTP paketů dle pravidel řetězce SLOWLORIS a rozhraní serveru tak není natolik vytíženo.



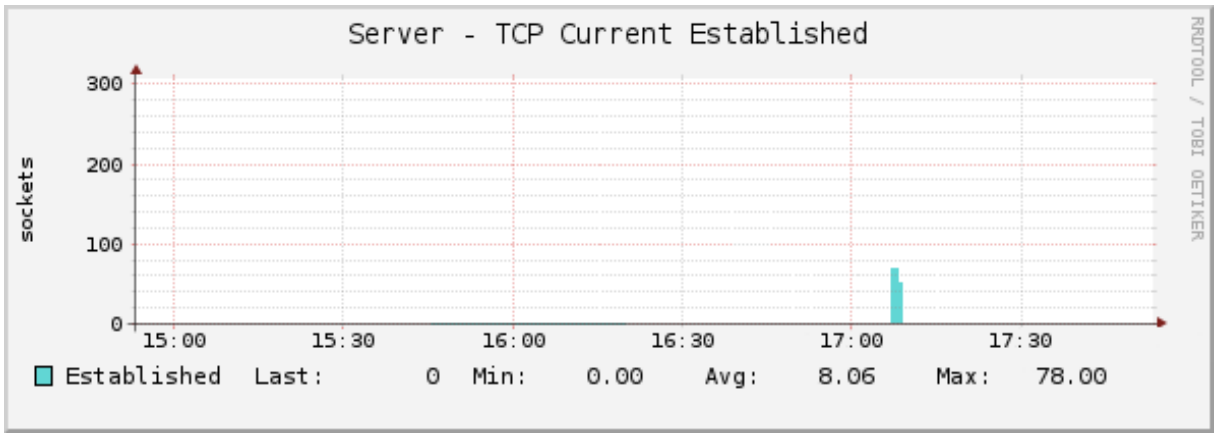
Obrázek 3.21: Počet příchozích HTTP Get/Error zpráv na server během útoku Slowloris na zabezpečenou síť.

Grafy zobrazující počet příchozích HTTP požadavků (obrázek 3.21) a stavy TCP spojení z pohledu serveru (obrázek 3.22) rovněž potvrzují úspěšné odvrácení slowloris útoku na server. Monitoring serveru zaznamenal průměrnou hodnotou počtu méně jak 100 příchozích HTTP Get požadavků na server během necelých 30 minut.



Obrázek 3.22: Počet a typ TCP paketů na serveru během Slowloris útoku na zabezpečenou síť.

Zachycená nejvyšší hodnota počtu 78 současně sestavených TCP spojení na serveru (obrázek 3.23) potvrzuje skutečnost, že se server nepotýkal s neobvykle vysokým počtem spojení a svou činnost tak vykonával bez jakýchkoliv známek omezení či nedostupnosti.



Obrázek 3.23: Počet sestavených TCP spojení na serveru během útoku Slowloris na zabezpečenou síť.

Dotazovaná webová stránka na serveru byla po dobu útoku zcela dostupná z pohledu paketového filtru, jen s minimálně zvýšenou dobou své odezvy. Tuto skutečnost přičítám použití virtualizačního prostředí, tedy systémovému vytížení hardwarových a softwarových prostředků hostitelského stroje během simulace DoS útoku, jenž nebyl nepatrný.

Závěr

V této diplomové práci se čtenář mohl seznámit s problematikou bezpečnosti počítačové sítě založené na použití frameworku Netfilter společně s nástrojem iptables. Hlavím cílem bylo vytvoření bezpečnostní politiky paketového filtru v prostředí Linux, která je schopná čelit vybraným a simulovaným síťovým útokům natolik, aby ochránila server v interní síti před nežádaným průnikem potencionálního útočníka.

První část práce se věnovala teorii pojednávající o základní charakteristice protokolu TCP/IP společně s popisem vývoje zařízení, které filtrují síťový provoz z jejich historického hlediska. Na základě popsaných vlastností jednotlivých typů paketových filtrů bylo zvoleno použití stavového paketového filtru, jako to prostředku k realizaci základní bezpečnostní politiky. Společně s detailním rozborem zvoleného frameworku Netfilter a jeho modulu iptables byla objasněna problematika použití souvisejících přípojných bodů, které jsou nedílnou součástí samotného procesu filtrování paketů. Tato teoretická část práce, seznamuje čtenáře rovněž s ideou a pravidly etického hackingu. Dále pak volbou linuxové distribuce Kali Linux, která nabízí možnost použití širokého spektra penetračních testů jako to jedné z metod hodnocení zabezpečení počítačových sítí. Byly objasněny související základní pojmy, rozdělení testů a metodologie testování. Závěrem kapitoly se čtenář seznámil s vybranými nástroji a jejich vlastnostmi v rámci Kali Linux, jenž byly Nping, Nmap a Wireshark.

V druhé části práce, která vychází z teoretických poznatkům předešlé části, jsem vytvořil základní bezpečnostní politiku stavového paketového filtru. Toho jsem docílil vytvořením skriptu, který obsahuje bezpečnostní pravidla, vycházejí z myšlenky použití serveru v rámci interní i externí virtualizované sítě. Disponuje pravidly souvisejícími se službami HTTP a HTTPS, jenž server nabízí uživatelům. Rovněž službu SSH určenou pro zabezpečený přístup k administraci samotného serveru společně se službou ICMP, použitou k monitorování dostupnosti. Tento samotný virtuální koncept měl za cíl přiblížit co nejvíce reálné prostředí, jako by byl server součástí veřejně dostupné sítě. Spuštěním skriptu došlo k aplikaci zvolených pravidel a následným testováním filtrace paketů prostřednictvím skenování portů protokolů TCP, UDP a ICMP. Z dosažených výsledků popsaných v rámci této kapitoly lze konstatovat, že zvolená bezpečností politika filtruje provoz tak, jak bylo očekáváno. Nežádoucí pakety iniciované útočníkem jsou blokovány a zaznamenávány do logovacího souboru paketového filtru.

Třetí stěžejní část práce, pojednává o charakteristikách útoků typu DoS a DDOS, jejich cílech a důvodech použití útočníky. Společně s popisem základního dělení DoS útoků jsou čtenáři přiblíženy svým popisem charakteristických vlastností a principů vybrané útoky typu ICMP flood, SSH Brute force a Slowloris. Před samotnými simulacemi jednotlivých útoků na server byl čtenáři objasněn postup, jakým jsem se při postupném vyhodnocování zaznamenaných výsledků ubíral.

Během v pořadí prvního ICMP flood útoku na nezabezpečenou síť pomocí nástroje Hping3, byl zaznamenán skokový nárůst počtu téměř dvou milionů příchozích ICMP Echo Request zpráv na server po dobu více jak devadesáti minut. V porovnání s nepravidelným počtem řádu desítek odeslaných zpráv typu ICMP Echo Reply tak server není schopen adekvátně reagovat na příchozí požadavky a stává se tak cílem útoku. Aplikací cílených pravidel pro filtraci ICMP protokolu a opakováním simulace útoku nyní již však na zabezpečenou síť, došlo ke znatelnému zlepšení situace z pohledu serveru v počtu příchozích ICMP zpráv. Zobrazené hodnoty z grafů poukazují na snížené

množství ICMP Echo Request zpráv a to včetně znatelné redukce datového toku v řádech desetin Mbit/s oproti maximální zaznamenané hodnotě téměř 6 Mbit/s v případě nezabezpečené sítě. Z výsledných grafů sledovaných údajů mohu konstatovat, že aplikací skriptu docílím filtraci ICMP protokolu na paketovém filtru dle požadavku. Server je z pohledu sítě dostupný i v případě působení útoku typu ICMP flood.

Simulace druhého útoku v pořadí, tedy SSH Brute force, spočívala v použití nástroje Ncrack. Útok na server skrze nezabezpečenou síť trval pouhých 84 sekund s tím, že program odhalil uživatelské jméno i heslo pro přístup na server. Probíhající útok byl zpozorován jak v záznamech logovacího souboru pro autentifikace serveru, tak i aktuálním počtem 227 SSH spojení z jedné IP adresy. Tato abnormalita lze jednoznačně označit za cílený útok na službu SSH serveru. Použitím skriptu s pravidly zamezující tento typ útoku a následným opakováním simulace na již zabezpečenou síť došlo k očekávaným reakcím paketového filtru. Zaznamenaný počet pouhých 4 příchozích požadavků na spojení z útočníkovy IP adresy odpovídá korektnímu nastavení pravidel. Tato hodnota potvrzuje společně s požadavkem na jedno spojení z IP adresy paketového filtru, který jsem během probíhajícího útoku inicioval, správnou funkčnost řetězců WHITELIST a BLACKLIST a dostupnost serveru i po dobu trvání útoku. Navrhnuté řešení k zabezpečení přístupu na server prostřednictvím služby SSH představuje díky dosaženým výsledkům funkční a přijatelnou ochranu.

Třetím a posledním útokem v pořadí, jsem simuloval útoky typu Slowloris. Útok, který si klade za cíl znepřístupnění webové služby Apache serveru prostřednictvím protokolu HTTP, potvrdil svou účinnost. Během simulovaného útoku na nezabezpečenou síť se webová stránka Apache serveru stala takřka okamžitě zcela nedostupnou jak z pohledu útočníka, tak samotného paketového filtru, a to po celou dobu probíhající simulace. Zaznamenanými abnormalitami v počtu příchozích 12,9 milionu HTTP Get zpráv, 5,7 milionu TCP SYN Received paketů, nárůstu datového toku na hodnotu téměř 12 Mbit/s na rozhraní serveru čí překročení počtu 256 současně možných navázaných TCP spojení nastaveným Apache v rámci monitorování specifických hodnot po dobu přibližně padesát minut lze jednoznačně rozpoznat indicie útoku typu HTTP flood. V porovnání s výsledky útoku na již zabezpečenou síť s cílenými filtračními pravidly je působení Slowloris útoku na server zanedbatelné. Dotazovaná webová stránka na serveru byla po dobu útoku zcela dostupná z pohledu paketového filtru, jen s minimálně zvýšenou dobou své odezvy. Tuto skutečnost přičítám použití virtualizačního prostředí, tedy systémovému vytížení hardwarových a softwarových prostředků hostitelského stroje během simulace DoS útoku, jenž nebyl nepatrný.

Z provedených simulací útoků, jejich souvisejících dopadů na server v nezabezpečené síti a následném vyhodnocení zaznamenaných výsledků působení na již zabezpečenou síť mohu konstatovat, že vytvořený skript bezpečnostní politiky splnil své očekávání a lze jej považovat za adekvátní ochranu před nežádanými průniky do sítě.

Použitá literatura

[1] Network security. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-12-10]. Dostupné z: en.wikipedia.org/wiki/Network_security

[2] Internet protocol suite. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-12-10]. Dostupné z: en.wikipedia.org/wiki/Internet_protocol_suite

[3] PUŽMANOVÁ, Rita. Moderní komunikační sítě od A do Z: [technologie pro datovou, hlasovou i multimediální komunikaci]. 2., aktualiz. vyd. Brno: Computer Press, 2006. ISBN 80-251-1278-

[4] STREBE, Matthew a Charles PERKINS. Firewally a proxy-servery: praktický průvodce. Brno: Computer Press, 2003. ISBN 80-7226-983-6.

[5] HONTAÑÓN, Ramón J. Linux: praktická bezpečnost. Praha: Grada, 2003. Profesional. ISBN 80-247-0652-0.

[6] DOSTÁLEK, Libor a Alena KABELOVÁ. Velký průvodce protokoly TCP/IP a systémem DNS. 2. aktualiz. vyd. Praha: Computer Press, 2000. Komunikace & sítě. ISBN 80-7226-323-4.

[7] De Schuymer, Bart a Fedchik , Nick. ebtables/iptables interaction on a Linux-based bridge. http://ebtables.netfilter.org/. [Online] netfilter.org, 9. November 2003. http://ebtables.netfilter.org/br_fw_ia/br_fw_ia.html#section1.

[8] BAUTTS, Tony, Terry DAWSON a GREGOR N. PURDY. Linux network administrator's guide: [infrastructure, service, and security]. 3. ed. Beijing: O'Reilly, 2005. ISBN 9780596005481.

[9] Linux. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-12-10]. Dostupné z: cs.wikipedia.org/wiki/Linux

[10] Debian. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-12-10]. Dostupné z: cs.wikipedia.org/wiki/Debian

[11] Virtualizace. In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-12-10]. Dostupné z: cs.wikipedia.org/wiki/Virtualizace

[12] Cacti (software). In: Wikipedia: the free encyclopedia [online]. San Francisco (CA): Wikimedia Foundation, 2017 [cit. 2017-03-10]. Dostupné z: https://en.wikipedia.org/wiki/Cacti_(software)

[13] ZARGAR, Saman Taghavi; JOSHI, James; TIPPER, David. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. IEEE Communications Surveys & Tutorials [online]. 2013, roc. 15, [cit. 2015-01-28], s. 2046–2069. Dostupné z WWW: <http://d-scholarship.pitt.edu/19225/1/ FinalVersion.pdf>. ISSN 1553-877X.

[14] SELECKÝ, Matúš. Penetrační testy a exploitace. Brno: Computer Press, 2012. ISBN 978-80-251-3752-9.

[15] Kali Linux. Kali Linux | Penetration Testing and Ethical Hacking Linux Distribution [online]. 2017 [cit. 2017-01-28]. Dostupné z: http://www.kali.org

[16] TOXEN, Bob. Bezpečnost v Linuxu: prevence a odvracení napadení systému. Brno: Computer Press, 2003. Security (CP Books). ISBN 80-7226-716-7.

[17] BARRETT, Daniel J., Richard E. SILVERMAN a Robert G. BYRNES. Linux security cookbook. Sebastopol, CA: O'Reilly, 2003. ISBN 978-0596003913.

[18] SMITH, Peter G. Linux network security. Hingham, Mass.: Charles River Media, c2005. ISBN 978-1584503965.